PowerBase Embedded Controller Installation and Use

VMEPBA/IH1

Notice

While reasonable efforts have been made to assure the accuracy of this document, Motorola, Inc. assumes no liability resulting from any omissions in this document, or from the use of the information obtained therein. Motorola reserves the right to revise this document and to make changes from time to time in the content hereof without obligation of Motorola to notify any person of such revision or changes.

No part of this material may be reproduced or copied in any tangible medium, or stored in a retrieval system, or transmitted in any form, or by any means, radio, electronic, mechanical, photocopying, recording or facsimile, or otherwise, without the prior written permission of Motorola, Inc.

It is possible that this publication may contain reference to, or information about Motorola products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that Motorola intends to announce such Motorola products, programming, or services in your country.

Restricted Rights Legend

If the documentation contained herein is supplied, directly or indirectly, to the U.S. Government, the following notice shall apply unless otherwise agreed to in writing by Motorola, Inc.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

Motorola, Inc.
Computer Group
2900 South Diablo Way
Tempe, Arizona 85282

Preface

The *PowerBase Embedded Controller Installation and Use* manual provides information you will need to install and use your PowerBase module, one of the MVME130*x* family of PCI Mezzanine Card (PMC) carrier boards.

MVME1301	8MB DRAM, no parity
MVME1302	8MB DRAM, parity
MVME1305	16MB DRAM, no parity
MVME1306	16MB DRAM, parity

The manual includes hardware preparation and installation instructions, information about using the front panel, a functional description, information about programming the board, using the PPCBug debugging firmware, and advanced debugger topics. Additional manuals you may wish to obtain are listed in Appendix A, *Ordering Related Documentation*. Other appendices provide the PowerBase specifications, connector pin assignments, and a glossary of terms.

This manual is intended for anyone who wants to design OEM systems, supply additional capability to an existing compatible system, or work in a lab environment for experimental purposes. A basic knowledge of computers and digital logic is assumed.

The PowerBase boards may be populated with a number of plug-together components, including PMCs, PROM mezzanine, transition board, and Serial Interface Modules (SIMs). Combinations of PowerBase boards and specific components are collectively referred to as *PowerCom*.

The information in this manual applies principally to the MVME130*x* PowerBase boards. The PMCs, PROM mezzanine, transition modules, and SIMs are described briefly here but are documented in detail in separate publications.

Document Terminology

Throughout this manual, a convention is used which precedes data and address parameters by a character identifying the numeric format as follows:

$	Dollar	Specifies a hexadecimal character
0x	Zero-x	
%	Percent	Specifies a binary number
&	Ampersand	Specifies a decimal number

For example, “12” is the decimal number twelve, and “$12” (hexadecimal) is the equivalent of decimal number eighteen. Unless otherwise specified, all address references are in hexadecimal.

An asterisk (*) following the signal name for signals which are level-significant denotes that the signal is true or valid when the signal is low.

An asterisk (*) following the signal name for signals which are edge-significant denotes that the actions initiated by that signal occur on high-to-low transition.

In this manual, *assertion* and *negation* are used to specify forcing a signal to a particular state. In particular, *assertion* and *assert* refer to a signal that is active or true; *negation* and *negate* indicate a signal that is inactive or false. These terms are used independently of the voltage level (high or low) that they represent.

Data and address sizes are defined as follows:

Byte	8 bits, numbered 0 through 7, with bit 0 being the least significant.
Half word	16 bits, numbered 0 through 15, with bit 0 being the least significant.
Word	32 bits, numbered 0 through 31, with bit 0 being the least significant.
Double word	64 bits, numbered 0 through 63, with bit 0 being the least significant.

Safety Summary
Safety Depends On You

The following general safety precautions must be observed during all phases of operation, service, and repair of this equipment. Failure to comply with these precautions or with specific warnings elsewhere in this manual violates safety standards of design, manufacture, and intended use of the equipment. Motorola, Inc. assumes no liability for the customer's failure to comply with these requirements.

The safety precautions listed below represent warnings of certain dangers of which Motorola is aware. You, as the user of the product, should follow these warnings and all other safety precautions necessary for the safe operation of the equipment in your operating environment.

Ground the Instrument.

To minimize shock hazard, the equipment chassis and enclosure must be connected to an electrical ground. The equipment is supplied with a three-conductor ac power cable. The power cable must be plugged into an approved three-contact electrical outlet. The power jack and mating plug of the power cable meet International Electrotechnical Commission (IEC) safety standards.

Do Not Operate in an Explosive Atmosphere.

Do not operate the equipment in the presence of flammable gases or fumes. Operation of any electrical equipment in such an environment constitutes a definite safety hazard.

Keep Away From Live Circuits.

Operating personnel must not remove equipment covers. Only Factory Authorized Service Personnel or other qualified maintenance personnel may remove equipment covers for internal subassembly or component replacement or any internal adjustment. Do not replace components with power cable connected. Under certain conditions, dangerous voltages may exist even with the power cable removed. To avoid injuries, always disconnect power and discharge circuits before touching them.

Do Not Service or Adjust Alone.

Do not attempt internal service or adjustment unless another person capable of rendering first aid and resuscitation is present.

Use Caution When Exposing or Handling the CRT.

Breakage of the Cathode-Ray Tube (CRT) causes a high-velocity scattering of glass fragments (implosion). To prevent CRT implosion, avoid rough handling or jarring of the equipment. Handling of the CRT should be done only by qualified maintenance personnel using approved safety mask and gloves.

Do Not Substitute Parts or Modify Equipment.

Because of the danger of introducing additional hazards, do not install substitute parts or perform any unauthorized modification of the equipment. Contact your local Motorola representative for service and repair to ensure that safety features are maintained.

Dangerous Procedure Warnings.

Warnings, such as the example below, precede potentially dangerous procedures throughout this manual. Instructions contained in the warnings must be followed. You should also employ all other safety precautions which you deem necessary for the operation of the equipment in your operating environment.

WARNING

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

WARNING **This equipment generates, uses, and can radiate electro-magnetic energy. It may cause or be susceptible to electro-magnetic interference (EMI) if not installed and used in a cabinet with adequate EMI protection.**

CE European Notice: Board products with the CE marking comply with the EMC Directive (89/336/EEC). Compliance with this directive implies conformity to the following European Norms:

EN55022 (CISPR 22)	Radio Frequency Interference
EN50082-1 (IEC801-2, IEC801-3, IEEC801-4)	Electromagnetic Immunity

The product also fulfills EN60950 (product safety) which is essentially the requirement for the Low Voltage Directive (73/23/EEC).

This board product was tested in a representative system to show compliance with the above mentioned requirements. A proper installation in a CE-marked system will maintain the required EMC/safety performance.

All Motorola PWBs (printed wiring boards) are manufactured by UL-recognized manufacturers, with a flammability rating of 94V-0.

The computer programs stored in the Read Only Memory of this device contain material copyrighted by Motorola Inc., 1995, and may be used only under a license such as those contained in Motorola's software licenses.

The software described herein and the documentation appearing herein are furnished under a license agreement and may be used and/or disclosed only in accordance with the terms of the agreement.

The software and documentation are copyrighted materials. Making unauthorized copies is prohibited by law.

No part of the software or documentation may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means without the prior written permission of Motorola, Inc.

Motorola® and the Motorola symbol are registered trademarks of Motorola, Inc.
PowerPC 603™ is a trademark of International Business Machines Corporation.
PowerPC™ is a trademark of International Business Machines Corporation and is used by Motorola with permission.
All other products mentioned in this document are trademarks or registered trademarks of their respective holders.

© Copyright Motorola 1996
All Rights Reserved
Printed in the United States of America
May 1996

Preparing and Installing the PowerBase 1

Introduction

This manual provides general information, hardware preparation and installation instructions, operating instructions, and a functional description of the PowerBase general purpose VMEbus board, based on the PowerPC™ 603 microprocessor. PowerBase is a powerful, low-cost embedded VME controller, as well as an intelligent Peripheral Component Interconnect (PCI) Mezzanine Card (PMC) carrier board. One of the primary applications for the PowerBase board is as a Wide Area Network (WAN) Controller module for the telecommunications market worldwide.

PowerBase Architecture

PowerBase consists of the base board, which may be an MVME1301, MVME1302, MVME1305, or MVME1306. These modules include support circuitry such as parity DRAM, PROM/Flash memory, and bridges to the Industry Standard Architecture (ISA) bus and the VMEbus.

PowerBase's PMC carrier architecture allows flexible configuration options and easy upgrades. It is designed to support one optional add-on PROM mezzanine module plus two PMCs in a single VMEmodule slot.

Optional PROM mezzanine board contains sockets for 4MB of One-Time-Programmable (OTP) PROM, organized as 64 bits wide.

Optional PMC products include capabilities such as Asynchronous Transfer Mode (ATM), Fiber Distributed Data Interface (FDDI), Fast Ethernet, Fast-Wide Small Computer System Interface (SCSI-2), and high-speed serial connectivity.

The PowerBase board supports front panel and P2 I/O on PMC slot 1, and front panel I/O only on PMC slot 2. Additionally, the 64 pins of I/O from PMC slot 1 are routed directly to P2. When

utilizing P2 I/O, it is necessary for customers to design their own connectivity/transition scheme.

Note Motorola offers a transition board, the MVME762, for use with the PowerBase. However, this board was designed to support only one configuration of a PowerBase/PMC combination: the ***PowerCom*** (MVME130*x* with the MPMC282-01*x* MC68360 PMC installed). For more details on this transition board, refer to the MVME762 and PowerCom documentation listed in Appendix A.

Figure 1-1 shows the PowerBase board and the positions of the PMCs and PROM mezzanine. The block diagram in Figure 1-2 illustrates the architecture of the MVME130*x* base board.

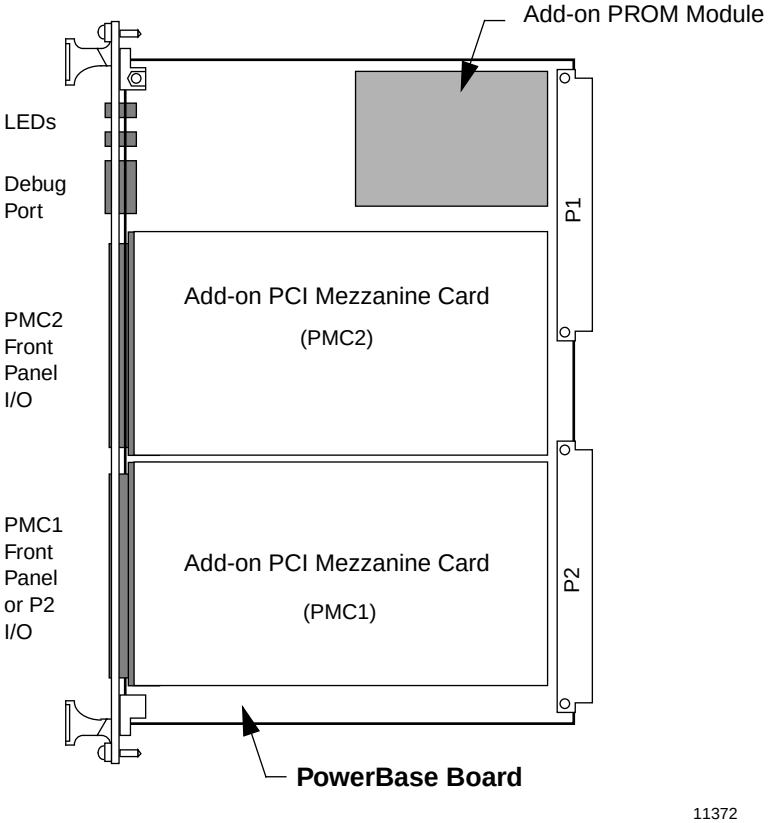


Figure 1-1. PowerBase with Optional PROM and PMC Mezzanines

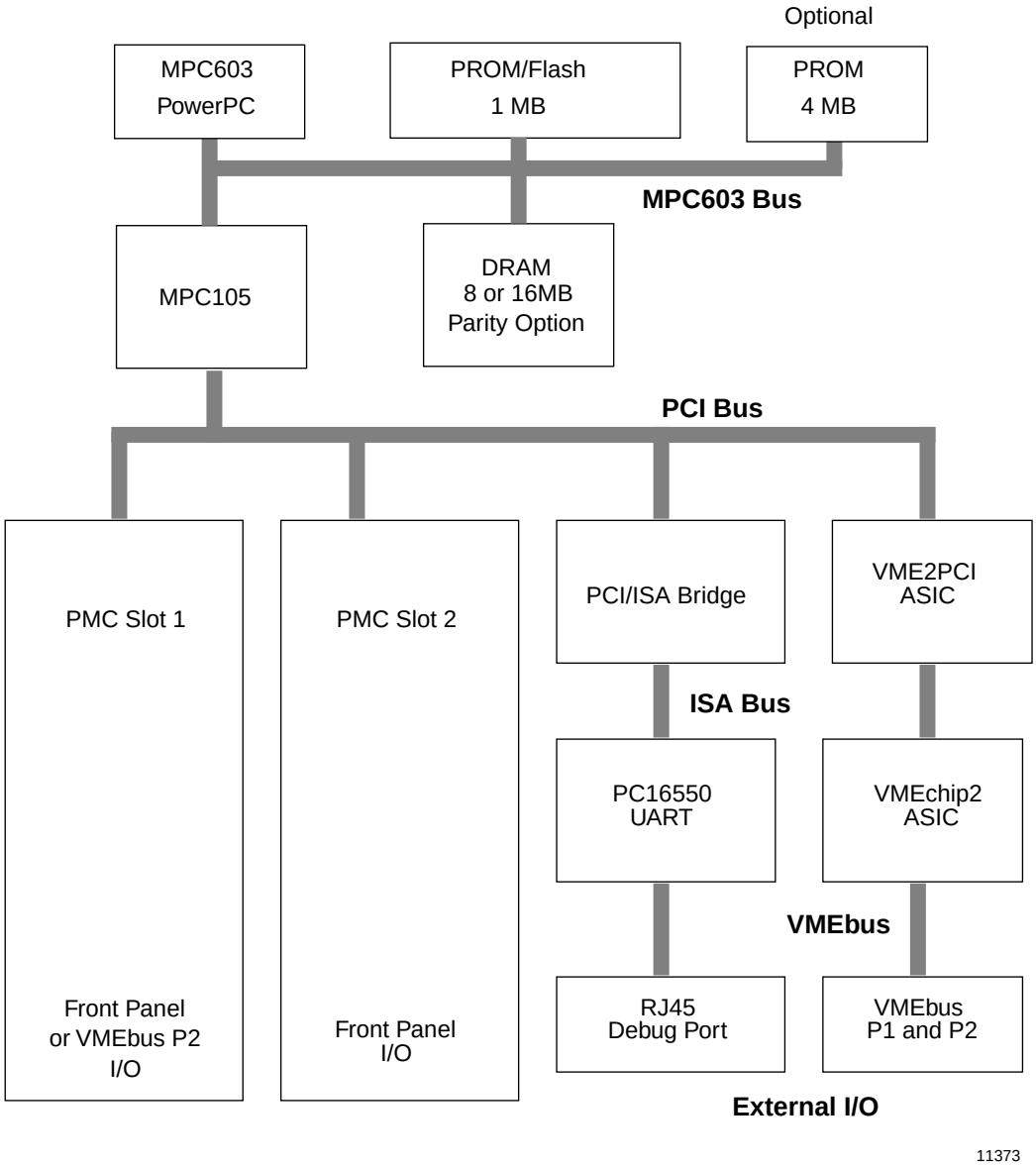


Figure 1-2. PowerBase Board Block Diagram

PowerBase interfaces to the VMEbus via the P1 and P2 connectors. It also draws +5V, +12V, and -12V power from the VMEbus backplane through these two connectors. The +3.3V power, used for the processor and PCI bridge chip and possibly for the PMC mezzanine, is derived onboard from the +5V power.

Support for two IEEE P1386.1 PCI mezzanine cards is provided via five 64-pin SMT connectors. Front panel openings are provided on the PowerBase board for the two PMC slots. In addition, there are 64 pins of I/O from PMC slot 1 that are routed to P2. The two PMC slots may contain two single width PMCs or one double width PMC.

Equipment Required

The following equipment is required to utilize a PowerBase board:

- MVME130*x* base board (PowerBase)
- VMEsystem enclosure

The following equipment is optional:

- PROM mezzanine module

Overview of Start-Up Procedures

The following table lists the things you will need to do before you can use this board, and tells where to find the information you need to perform each step. Be sure to read this entire chapter and read all Caution and Warning notes before beginning.

Table 1-1. Start-Up Overview

What you need to do ...	Refer to ...	On page ...
Unpack the hardware.	***Unpacking the Hardware***	1-7
Configure the hardware by setting jumpers on the board.	***Configuring the Hardware***	1-7
	Preparing the PowerBase	1-8
Install optional PMC(s), if any.	***Installing Optional PMCs***	1-17
	PMC Slots	2-4
	For additional information on PMCs, refer to the documentation furnished with the PMCs.	
Connect a console terminal.	***Installing the Hardware***	1-13
	Debug Port	2-3
	Installation Considerations	1-18
Connect any other optional devices or equipment you will be using.	***Connector Pin Assignments***	C-1
	For more information on optional devices and equipment, refer to the documentation provided with that equipment.	
Power up the system.	***Installing the Hardware***	1-13
	Status Indicators (DS1 - DS4)	2-2
	If any problems occur, refer to the section ***Performing Diagnostic Tests*** in Chapter 5, ***Using PPCBug***.	5-17
	You may also wish to obtain the ***PPC1Bug Diagnostics Manual***, listed in the table entitled ***Motorola Computer Group Documents*** in Appendix A, ***Ordering Related Documentation***.	

Table 1-1. Start-Up Overview (Continued)

What you need to do ...	Refer to ...	On page ...
Change any environmental parameters as you wish.	***ENV - Set Environment***	6-2
	You may also wish to obtain the ***PPCBug Firmware Package User's Manual***, listed in the table ***Motorola Computer Group Documents*** in Appendix A, ***Ordering Related Documentation***.	
Install the optional PROM mezzanine, if desired (NOTE: power must be off).	***Installing an Optional PROM Mezzanine***	1-15
Program the MVME130*x* module as needed for your applications.	***Configuring the Hardware***	1-7
	Programming the PowerBase	4-1
	You may also wish to obtain the ***PowerBase Embedded Controller Programmer's Reference Guide***, listed in the table ***Motorola Computer Group Documents*** in Appendix A, ***Ordering Related Documentation***.	

Unpacking the Hardware

Note If the shipping carton is damaged upon receipt, request that the carrier's agent be present during the unpacking and inspection of the equipment.

Unpack the equipment from the shipping carton. Refer to the packing list and verify that all items are present. Save the packing material for storing and reshipping of equipment.

Caution Avoid touching areas of integrated circuitry; static discharge can damage these circuits.

Configuring the Hardware

To produce the desired configuration and ensure proper operation of the PowerBase board, you may need to carry out certain modifications before and after installing the module.

The MVME130*x* provides software control over most options; by setting bits in control registers after installing the MVME130*x* in a system, you can modify its configuration. The MVME130*x* control registers are described in Chapter 4, with additional information in the *PowerBase Embedded Controller Programmer's Reference Guide* as listed in table ***Motorola Computer Group Documents*** in Appendix A, ***Ordering Related Documents***.

Some options, however, are not software-programmable. Such options are controlled through manual installation or removal of header jumpers or interface modules on the base board or the associated modules.

Serial ports on the optional PMC boards are manually configurable. For a discussion of the configurable items on the PMCs, refer to the user's manual for the particular PMC.

Preparing the PowerBase

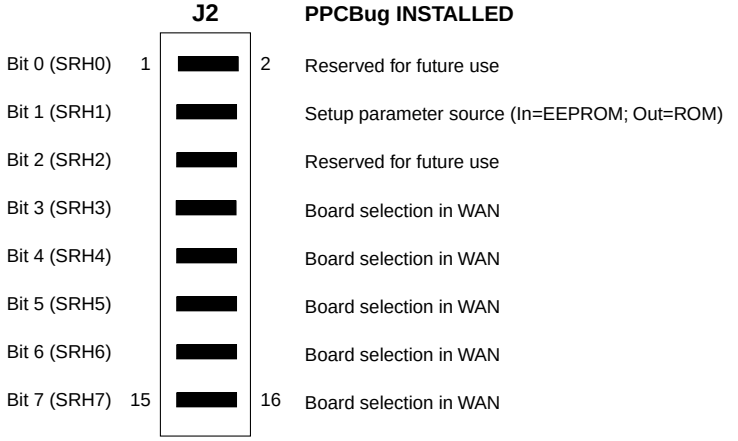
Figure 1-3 illustrates the placement of the switches, jumper headers, connectors, and LED indicators on the MVME130*x*. Manually configurable items on the base board include:

- General-purpose software-readable header (J2)
- VMEbus system controller selection (J5)

The MVME130*x* has been factory tested and is shipped with the configurations described in the following sections. The MVME130*x* factory-installed debug monitor, PPCBug, operates with those factory settings.

Setting the General-Purpose Software-Readable Header (J2)

Header J2 provides eight readable jumpers. These jumpers can be read as a register at ISA I/O address $801 (hexadecimal). Bit 0 is associated with header pins 1 and 2; bit 7 is associated with pins 15 and 16. The bit values are read as a **0** when the jumper is installed, and as a **1** when the jumper is removed. The MVME130*x* is shipped from the factory with J2 set to all **0**s (jumpers on all pins), as shown in the diagram below.



The PowerPC firmware, PPCBug, reserves all bits, SRH0 to SRH7.

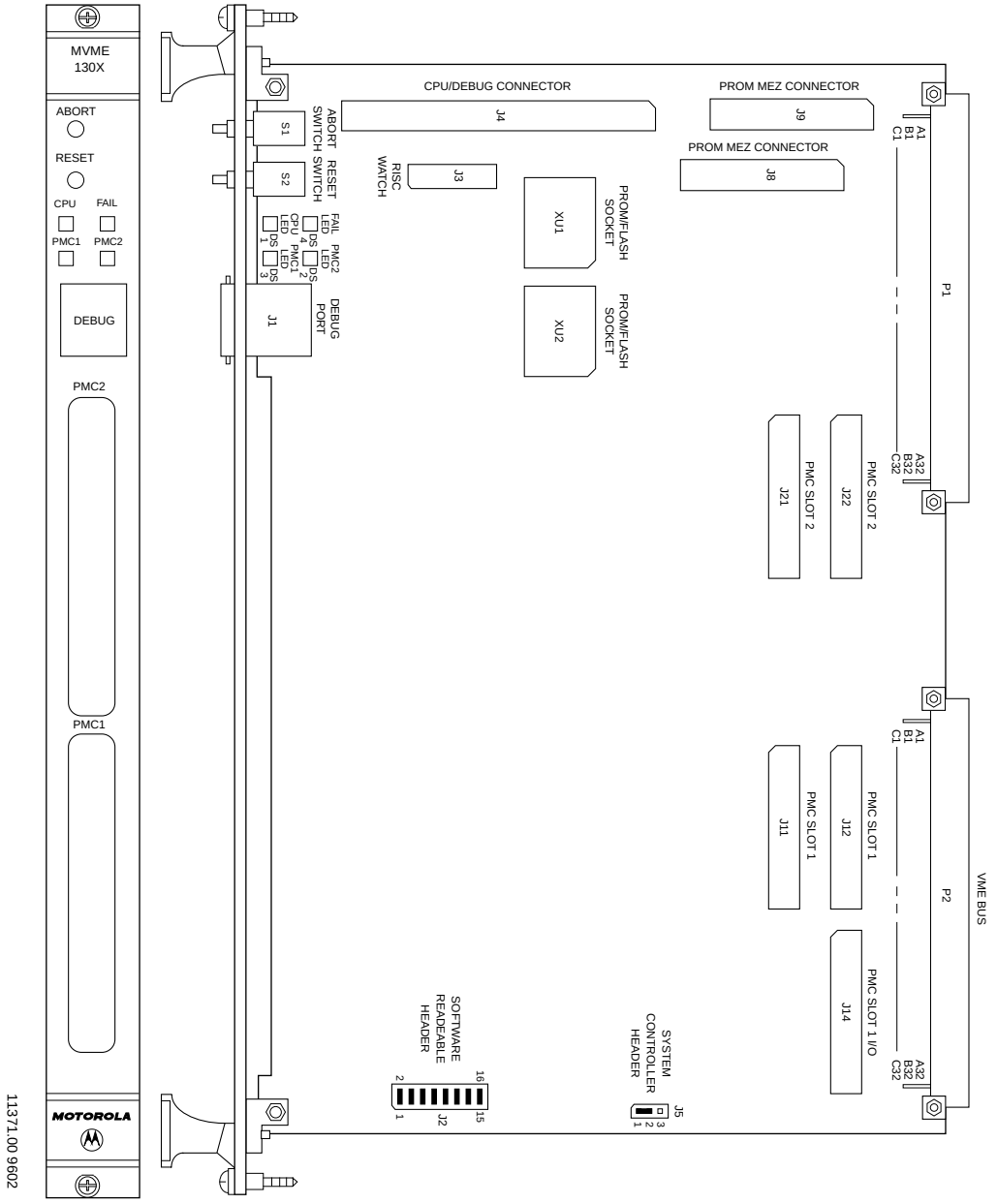


Figure 1-3. MVME130*x* Switches, LEDs, Headers, Connectors

With the jumper installed between pins 3 and 4 (factory configuration), the debugger uses the current user setup/operation parameters in EEPROM. When the jumper is removed (making the bit a **1**), the debugger uses the default setup/operation parameters in ROM instead. Refer to the **ENV** command description in Chapter 6 for the ROM defaults.

The five higher-order bits, SRH3 to SRH7, are required to get GCSR locations for up to 18 boards when used in a Wide Area Network (WAN). The MVME130*x* is shipped from the factory with J2 set to all **0**s (jumpers on all pins); each PowerBase board in the system must have a unique address in order for the system to function correctly. These addresses are given in Table 1-2.

There is a method to issue commands to the PPCBug on the PowerBase board via the VMEbus. This method utilizes the Global Control and Status Registers (GCSR) in the VMEchip2. Eight 16-bit registers are included in the GCSR set. These registers are accessible in the VMEbus short I/O space responding to address modifier codes $29 or $2D. The specific address of the GCSR on the VMEbus is programmed in the VMEchip2 LCSR by the debugger and is determined by the conditions described below.

At start-up, PPCBug, when executing on PowerBase, reads the state of the upper five jumpers on the software readable header J2. These jumpers determine a board select number. Each board select number has been assigned a specific group and board address in VMEbus short I/O space, as shown in Table 1-2.

The specific assigned address is written into the LCSR and EEPROM unless all five jumpers have been removed (board select number 31). If all five jumpers are removed, then the values previously stored in EEPROM for group and board addresses are programmed in the LCSR. Values may be stored in the EEPROM by setting the board select number with jumpers on J2, or with the debugger **ENV** command. In this way, any jumper setting a board select number other than 31 will over-ride and set the GCSR address, yet any VMEbus short I/O address may be set using the **ENV** command and the board select set to 31.

Table 1-2. GCSR Addresses

Board Select Number	GCSR Address	J2 Jumper Setting	Notes
0	$FFFFD800	XXXIIIII	1, 2
1	$FFFFD810	XXX:IIII	1
2	$FFFFD820	XXXI:III	1
3	$FFFFD830	XXX::III	1
4	$FFFFD840	XXXII:II	1
5	$FFFFD850	XXX:I:II	1
6	$FFFFD860	XXXI::II	1
7	$FFFFD870	XXX:::II	1
8	$FFFFD880	XXXIII:I	1
9	$FFFFD890	XXX:II:I	1
10	$FFFFD8A0	XXXI:I:I	1
11	$FFFFD8B0	XXX::I:I	1
12	$FFFFD8C0	XXXII::I	1
13	$FFFFD8D0	XXX:I::I	1
14	$FFFFD8E0	XXXI:::I	1
Note: $FFFFD8F0 is not used.			
15	$FFFFD900	XXX::::I	1
16	$FFFFD910	XXXIIII:	1
17	$FFFFD920	XXX:III:	1
18	$FFFFD930	XXXI:II:	1
Note: $FFFFD800 is used for all others			
31	From EEPROM	XXX:::::	1

Notes

1. The first three jumper positions, for bits SRH0 through 2, may have unrelated functions. This is represented above as **XXX**. An **I** = jumper installed on the pins; a **:** = no jumper on the pins.

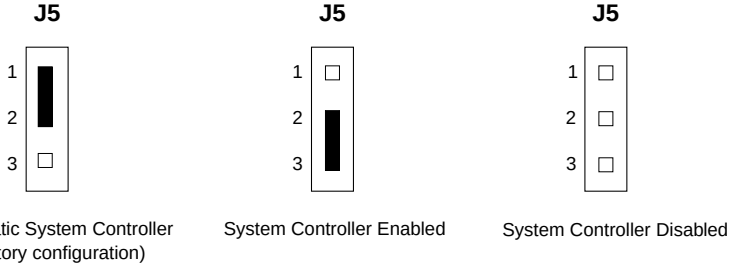
2. Default setting.

Setting the VMEbus System Controller Selection Header (J5)

The MVME130*x* is factory-configured in automatic system controller mode; i.e., a jumper is installed across pins 1 and 2 of header J5. This means that the MVME130*x* determines if it is system controller at system power-up or reset by its position on the bus; if it is in slot 1 on the VME system, it configures itself as the system controller.

Install the jumper across pins 2 and 3 if you intend to operate the MVME130*x* as system controller in all cases.

Remove the jumper from J5 if the MVME130*x* is not to operate as system controller under any circumstances.



Automatic System Controller (factory configuration)

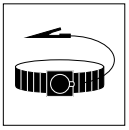
System Controller Enabled

System Controller Disabled

Installing the Hardware

The following paragraphs discuss the installation of the PowerBase into a VME chassis, and installing the optional PMC boards and PROM mezzanine board onto the PowerBase.

Taking ESD Precautions



Motorola strongly recommends that you use an antistatic wrist strap and a conductive foam pad when installing or upgrading a system. Electronic components, such as disk drives, computer boards, and memory modules, can be extremely sensitive to Electro-Static Discharge (ESD). After removing the component from the system or its protective wrapper, place the component flat on a grounded, static-free surface (and in the case of a board, component side up). Do not slide the component over any surface.

If an ESD station is not available, you can avoid damage resulting from ESD by wearing an antistatic wrist strap (available at electronics stores) that is attached to an unpainted metal part of the system chassis.

Installing the PowerBase

Before installing the PowerBase into your VME chassis, ensure that the jumpers on PowerBase's J2 and J5 headers are configured, as previously described. If you intend to mount the optional PCI mezzanine cards (PMCs) or PROM mezzanine on your PowerBase, refer to the sections ***Installing Optional PMCs*** and ***Installing the Optional PROM Mezzanine*** before you begin this procedure.

Proceed as follows to install the PowerBase in the VME chassis:

1. Attach an ESD strap to your wrist. Attach the other end of the ESD strap to the chassis as a ground. The ESD strap must be secured to your wrist and to ground throughout the procedure.
2. Perform an operating system shutdown:
   a. Turn the AC or DC power off and remove the AC cord or DC power lines from the system.

Caution

Inserting or removing modules with power applied may result in damage to module components.

Warning

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

b. Remove chassis or system cover(s) as necessary for access to the VMEmodules.

3. Remove the filler panel from the card slot where you are going to install the PowerBase board.
   – If you intend to use the MVME130*x* as system controller, it must occupy the leftmost card slot (slot 1). The system controller must be in slot 1 to correctly initiate the bus-grant daisy-chain and to ensure proper operation of the IACK daisy-chain driver.
   – If you do not intend to use the MVME130*x* as system controller, it can occupy any unused card slot.

Caution

Avoid touching areas of integrated circuitry; static discharge can damage these circuits.

4. Slide the MVME130*x* into the selected card slot. Be sure the module is seated properly in the P1 and P2 connectors on the backplane. Do not damage or bend connector pins.
5. Secure the MVME130*x* in the chassis with the screws provided, making good contact with the transverse mounting rails to minimize RF emissions.

Note Some VME backplanes (e.g., those used in Motorola "Modular Chassis" systems) have an auto-jumpering feature for automatic propagation of the IACK and BG signals. Step 6 does not apply to such backplane designs.

6. On the chassis backplane, remove the INTERRUPT ACKNOWLEDGE (IACK) and BUS GRANT (BG) jumpers from the header for the card slot occupied by the MVME130*x*.
7. If you intend to use PPCBug interactively, connect the terminal that is to be used as the PPCBug system console to the DEBUG port on the front panel of the PowerBase. Set up the terminal as follows:
   - Eight bits per character
   - One stop bit per character
   - Parity disabled (no parity)
   - Baud rate = 9600 baud (default baud rate of the port at power-up)

   In normal operation the host CPU controls PowerBase operation via the VMEbus GCSR registers (refer to *MultiProcessor Support* in Chapter 5). In normal operation, connection of a debug console terminal is required only if you intend to use PPCBug interactively.
8. Replace the chassis or system cover(s), cable peripherals to the panel connectors as appropriate, reconnect the system to the AC or DC power source, and turn the equipment power on.
9. The MVME130x's **CPU** LED indicates activity as a set of confidence tests is run, and the debugger prompt `PPC1-Bug>` appears.

Installing the Optional PROM Mezzanine

Caution

When you install the PROM Mezzanine on your PowerBase, the Flash devices will be inaccessible, and you will not be able to use the PPCBug functions in Flash.

Proceed as follows to install the optional PROM mezzanine on your PowerBase board:

1. Attach an ESD strap to your wrist. Attach the other end of the ESD strap to the chassis as a ground. The ESD strap must be secured to your wrist and to ground throughout the procedure.

Note If your MVME130*x* has not yet been installed in a chassis, go to step 4.

2. Perform an operating system shutdown:
   a. Turn the AC or DC power off and remove the AC cord or DC power lines from the system.

Caution

Inserting or removing modules with power applied may result in damage to module components.

Warning

Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

   b. Remove chassis or system cover(s) as necessary for access to the VMEmodules.

3. Remove the screws securing the MVME130*x* to the chassis and remove the MVME130*x* from its card slot.
4. Lay the MVME130*x* on a level surface with the P1 and P2 connectors on your right, and position the PROM mezzanine above it in the upper right-hand corner. (Refer to Figure 1-1.)

Caution

Avoid touching areas of integrated circuitry; static discharge can damage these circuits.

5. Gently seat the PROM mezzanine onto the PowerBase module, ensuring that the connectors are properly aligned. (Refer to Figure 1-3.)
6. Using the provided screws, fasten the PROM mezzanine to the MVME130*x*.
7. Reinstall the MVME130*x* board in the chassis according to the instructions given in the *Installing the PowerBase* section of this chapter.

Installing Optional PMCs

Proceed as follows to install optional PCI mezzanine cards (PMCs) on your PowerBase board:

1. Attach an ESD strap to your wrist. Attach the other end of the ESD strap to the chassis as a ground. The ESD strap must be secured to your wrist and to ground throughout the procedure.

Note If your MVME130*x* has not yet been installed in a chassis, go to step 4.

2. Perform an operating system shutdown:
   a. Turn the AC or DC power off and remove the AC cord or DC power lines from the system.

Caution Inserting or removing modules with power applied may result in damage to module components.

Warning Dangerous voltages, capable of causing death, are present in this equipment. Use extreme caution when handling, testing, and adjusting.

b. Remove chassis or system cover(s) as necessary for access to the VMEmodules.

3. Remove the screws securing the MVME130*x* to the chassis and remove the MVME130*x* from its card slot.
4. Lay the MVME130*x* on a level surface with the P1 and P2 connectors closest to you, and position the PMC above it in the center (for PMC2) or left-hand side (for PMC1). (Refer to Figure 1-1.)

Caution

Avoid touching areas of integrated circuitry; static discharge can damage these circuits.

5. Insert the PMC's front panel bezel through the selected PMC slot in the PowerBase's front panel and gently seat the PMC onto the PowerBase, ensuring that the keying pin and connectors are properly aligned. (Refer to Figure 1-3.)
6. Turn the PowerBase over, and using the four provided screws, fasten the PMC to the MVME130*x* from the underside.
7. Reinstall the MVME130*x* board in the chassis according to the instructions given in the *Installing the PowerBase* section of this chapter.

Installation Considerations

The PowerBase board draws power from the VMEbus backplane connectors P1 and P2. P2 is also used for the upper 16 bits of data in 32-bit transfers, and for the upper 8 address lines in extended addressing mode. The MVME130*x* may not function properly without its main board connected to VMEbus backplane connectors P1 and P2.

Whether the PowerBase operates as a VMEbus master or as a VMEbus slave, it is configured for 32 bits of address and 32 bits of data (A32/D32). However, it handles A16 or A24 devices in the

address ranges indicated in Chapter 4. D8 and/or D16 devices in the system must be handled by the PowerPC processor software. Refer to the memory maps in Chapter 4.

The MVME130*x* contains shared onboard DRAM whose base address is software-selectable. Both the onboard processor and off-board VMEbus devices see this local DRAM at base physical address $00000000, as programmed by the PPCBug firmware. This may be changed via software to any other base address. Refer to the *PowerBase Programmer's Reference Guide* for more information.

If the MVME130*x* tries to access off-board resources in a nonexistent location and is not system controller, and if the system does not have a global bus time-out, the MVME130*x* waits forever for the VMEbus cycle to complete. This will cause the system to lock up. There is only one situation in which the system might lack this global bus time-out: when the MVME130*x* is not the system controller and there is no global bus time-out elsewhere in the system.

Multiple PowerBase boards may be installed in a single VME chassis. Each must have a unique GCSR address, selected by setting jumpers on its J2 header, as described in *Preparing the PowerBase*. In general, hardware multiprocessor features are supported.

Other MPUs on the VMEbus can interrupt, disable, communicate with, and determine the operational status of the processor(s). One register of the GCSR (global control/status register) set includes four bits that function as location monitors to allow one MVME130*x* processor to broadcast a signal to any other MVME130*x* processors. All eight registers are accessible from any local processor as well as from the VMEbus.

Using the Front Panel 2

Introduction

This chapter provides information about the switches, status indicators, and I/O ports on the front panel of the PowerBase. The front panel is pictured in Figure 1-3 in Chapter 1.

Switches

There are two switches (ABORT and RESET) and four LED (light-emitting diode) status indicators (CPU, FAIL, PMC1, PMC2) located on the MVME130*x* front panel.

ABORT (S1)

When activated by software, the ABORT switch can generate an interrupt signal from the base board to the processor at a user-programmable level. The interrupt is normally used to abort program execution and return control to the PPCBug debugger firmware located in the PowerBase Flash or PROM memory. The interrupt signal reaches the processor module via ISA bus interrupt line IRQ8*. The signal status is also available from the general purpose I/O port (refer to Chapter 4). This also allows software, after an IRQ8* interrupt, to poll the ABORT switch and verify that it has been pressed.

The interrupter connected to the ABORT switch is an edge-sensitive circuit, filtered to remove switch bounce.

RESET (S2)

The RESET switch resets all onboard devices and causes HRESET* to be asserted in the MPC603; it also drives a SYSRESET* signal if the PowerBase is the system controller.

The VMEchip2 includes both a global and a local reset driver. When the VMEchip2 operates as the VMEbus system controller, the reset

driver provides a global system reset by asserting the VMEbus signal SYSRESET*. A SYSRESET* signal may be generated by the RESET switch, a power-up reset, a watchdog time-out, or by a control bit in the LCSR in the VMEchip2. SYSRESET* remains asserted for at least 200 ms, as required by the VMEbus specification.

Similarly, the VMEchip2 provides an input signal and a control bit to initiate a local reset operation. By setting a control bit, software can maintain a board in a reset state, disabling a faulty board from participating in normal system operation. The local reset driver is enabled even when the VMEchip2 is not the system controller. A local reset may be generated by the RESET switch, a power-up reset, a watchdog time-out, a VMEbus SYSRESET*, or a control bit in the GCSR.

Status Indicators (DS1 - DS4)

There are four LED (light-emitting diode) status indicators located on the MVME130*x* front panel.: CPU, FAIL, PMC1, and PMC2.

CPU (DS1)

This *green* LED indicates CPU activity; lights when the DBB* (Data Bus Busy) signal line on the processor bus is active.

FAIL (DS4)

This *yellow* LED indicates board failure; lights when the BRDFAIL* signal line is active.

PMC1 (DS3)

This *green* LED indicates PCI activity; lights when the PCI bus grant to PMC1 signal line on the PCI bus is active. This indicates that a PMC (if installed) is active.

PMC2 (DS2)

This *green* LED indicates PCI activity; lights when the PCI bus grant to PMC2 signal line on the PCI bus is active. This indicates that a PMC (if installed) is active.

DEBUG Port

The RJ45 port labeled DEBUG on the front panel of the PowerBase supplies the PowerBase serial communications interface, implemented via a UART PC16550 controller chip from National Semiconductor. It is asynchronous only. This serial port is configured for EIA-232-D DTE, as shown in Figure 2-1.

The DEBUG port may be used for connecting a terminal to the PowerBase to serve as the firmware console for PowerBase's factory installed debugger, PPCBug. The port is configured as follows:

- 8 bits per character
- 1 stop bit per character
- Parity disabled (no parity)
- Baud rate = 9600 baud (default baud rate at power-up)

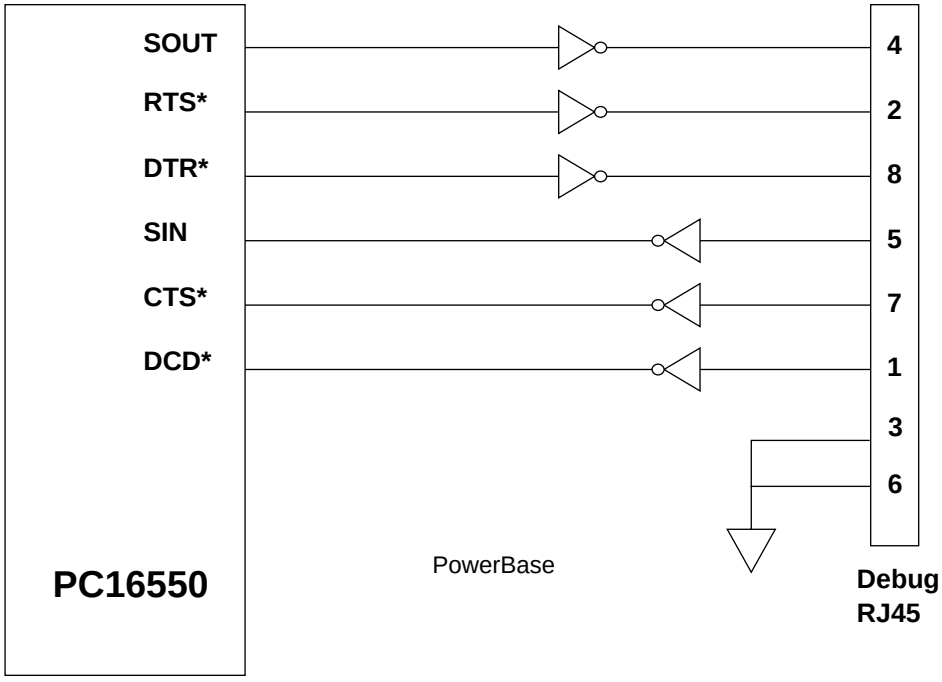


Figure 2-1. Debug Port Configuration

After power-up, the baud rate of the DEBUG port can be reconfigured by using the debugger's Port Format (**PF**) command. Refer to Chapters 5 and 6 for information about PPCBug.

PMC Slots

Two openings located on the front panel provide I/O expansion by allowing access to one or two single-wide or one double-wide PCI Mezzanine Card (PMC), connected to the PMC connectors on the PowerBase. For pin assignments for the PMC connectors, refer to Appendix C.

Warning

Do not attempt to install any PMC boards without performing an operating system shutdown and following the procedures given in Chapter 1.

For further information on PMCs, refer to the user's manual for the particular PMC.

PMC1 (PMC Slot 1)

The opening labeled PMC1 on the front panel provides I/O access to a PMC when it is connected to the 64-pin SMT connectors J11 and J12 on the PowerBase board.

When a PMC is connected to the 64-pin SMT connectors J11, J12, and J14 on the PowerBase board, 64 pins of I/O from PMC slot 1 are routed to P2, the 96-pin VMEbus connector on the back of the PowerBase board. Both P2 and the front panel are then accessible for I/O.

PMC2 (PMC Slot 2)

The opening labeled PMC2 on the PowerBase front panel provides I/O access to a PMC when it is connected to 64-pin SMT connectors J21 and J22 on the PowerBase board.

Double-Wide PMCs

A double-wide, 8-port PMC, connected to J11, J12, J21, and J22, would allow front panel I/O access through both PMC1 and PMC2, and optionally the P2 connector if J14 is also used.

Functional Description 3

Introduction

This chapter describes the PowerBase embedded controller on a block diagram level. The *General Description* provides an overview of the MVME130*x*, followed by a detailed description of several blocks of circuitry. Figure 3-1 shows a block diagram of the overall board architecture.

Detailed descriptions of other MVME130*x* blocks, including programmable registers in the ASICs and peripheral chips, can be found in the *PowerBase Embedded Controller Programmer's Reference Guide*. Refer to it for a functional description of the MVME130*x* in greater depth.

General Description

The PowerBase is a general purpose, high-performance, embedded controller VMEmodule based on the 66MHz PowerPC 603 microprocessor.

As shown in the *PowerBase Features* section, PowerBase offers many standard features desirable in a computer system, including a debug port, Boot ROM, PROM/Flash memory, DRAM, and interface for two PCI Mezzanine Cards (PMCs), contained in a one-slot VME package. Its flexible mezzanine architecture allows relatively easy upgrades of the I/O and/or memory.

There are four standard buses on PowerBase:

PowerPC 603 Processor Bus
PCI Local Bus
ISA Bus
VMEbus

As shown in Figure 3-1, an MPC105 PCI Bridge/Memory Controller provides the interface from the Processor Bus to PCI. An 82378ZB device performs the bridge function between PCI and ISA. Two ASIC devices, VME2PCI and VMEchip2, provide the interface between the PCI Local Bus and the VMEbus.

The PCI local bus is a key feature. In addition to the on-board local bus peripherals, the PCI bus supports an industry-standard mezzanine interface, IEEE P1386.1 PMC (PCI Mezzanine Card).

PowerBase Features

The base board contains the following:

- 66.66MHz MPC603 PowerPC™ processor
- 8 or 16MB of DRAM, with or without parity (build options)
- 1MB of onboard PROM/Flash memory for Boot ROM
- Option for 4MB of PROM on a mezzanine board
- VMEbus interface implemented with VMEchip2 ASIC
  - VMEbus system controller functions
  - VMEbus interface to local bus (A24/A32, D8/D16/D32/BLT(D8/D16/D32/D64))
  - Local bus to VMEbus interface (A16/A24/A32, D8/D16/D32)
  - VMEbus interrupter
  - VMEbus interrupt handler
  - Global CSR for interprocessor communications
  - DMA for fast local memory - VMEbus transfers (A16/A24/A32, D16/D32/D64).
- Interfaces to two P1386.1 PCI Mezzanine Cards (PMCs)
  - Accepts two single-width PMCs or one double-width PMC
  - Front panel and/or VMEbus P2 I/O on PMC slot 1

  - Front panel I/O on PMC slot 2

- One asynchronous serial port (debug port) via an RJ45 front panel connector
- 8-bit Software Readable Header
- 8KB of EEPROM
- RESET switch
- ABORT switch
- Status LEDs for FAIL, CPU, PMC1, and PMC2

Refer to Appendix B for product specifications.

PowerBase Components

Figure 3-1 is a block diagram of the MVME130*x*'s overall architecture.

RISCWatch Header

A 16-pin male 2x8 header is provided for connecting to the RISCWatch MPC603 processor interface. MPC603 interconnect information is provided in Appendix C.

PCI Bridge/Memory Controller (MPC105)

A Motorola MPC105 device provides the necessary interface between the MPC603 processor, the Flash memory, the PROM, the DRAM, and the PCI Local Bus. The MPC105 supports various PowerPC processor external bus frequencies up to 66.66MHz and PCI frequencies up to 33.33MHz. Table 3-1 summarizes the clock frequencies supported by the PowerBase:

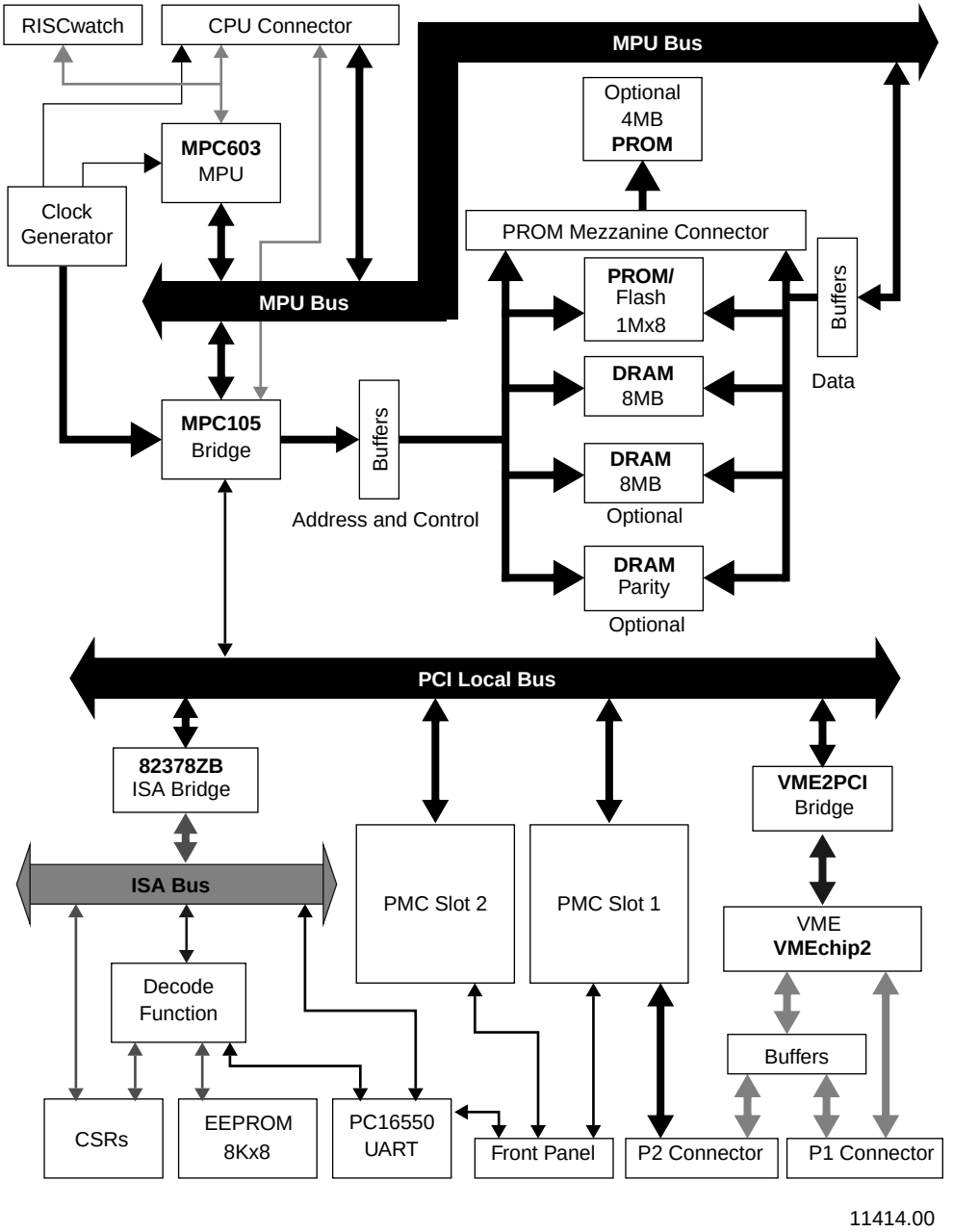


Figure 3-1. MVME130*x* Block Diagram

Table 3-1. MPC603 Clock Frequencies Supported by PowerBase

MPC603 Internal Processor Speed	MPC603 External Bus Speed	MPC603 Clock PLL	MPC105 Internal Clock Frequency	MPC105 External Clock Frequency	MPC105 Clock PLL	PCI Bus Frequency
66.66MHz	66.66MHz	1x	66.66MHz	33.33MHz	2x	33.33MHz

Flash Devices and Boot ROM

The PowerBase has two onboard 32-pin PLCC sockets which support two banks of 512Kx8 PROM or Flash devices that together provide a total of 1MB of PROM/Flash capacity. This memory space is provided for the Boot ROM function.

The factory configuration is with Flash devices installed. Since the devices are socketed, they are programmed externally before they are installed on the board. The onboard monitor/debugger, PPCBug, resides in the Flash devices. PPCBug provides:

- A boot loader and extensive on-board diagnostics
- A single-line assembler-disassembler
- A remote boot capability

Each bank of PROM/Flash is 8-bits wide and is controlled by the MPC105. The MPC105 performs byte alignment for write accesses and also packs bytes for 16-bit, 32-bit, and 64-bit read accesses to the PROM/Flash area.

The PROM/Flash must be accessed in the MPC105 non-burst mode. For PROM/Flash speed of 150ns, software should program ROMFAL (first access length) and ROMNAL (last access length) in the MPC105 device with the following values:

Table 3-2. Minimum ROMFAL and ROMNAL Values at 150ns

Processor External Bus Speed	Minimum ROMFAL	Minimum ROMNAL
66.66MHz	10	0

Note The onboard PROM/Flash memory is disabled when the PROM mezzanine is installed, because the MPC105 cannot support both memory interfaces simultaneously.

A status bit is available from the general purpose I/O port to allow the firmware to know whether it is operating from on-board PROM/Flash or from the PROM mezzanine.

Since Flash and PROM devices have different pinouts, the MVME130*x* must be hard-wired with zero-ohm resistors for Flash or PROM memory configuration. The factory configuration is for Flash. PowerBase supports these PROM/Flash device types:

Flash (default)	AM29F040-150JC (or equivalent)
PROM	AM27C040-150JC (or equivalent)

Caution As supplied, the two PLCC sockets hold Flash devices that contain bootstrap firmware. If these devices are removed or corrupted, your system will not boot. Replacement PROM/Flash devices must have board initialization and boot capability.

PROM/Flash Latency

The following table shows the PROM/Flash read latency at the processor/memory bus frequency:

Table 3-3. PROM/Flash Read Latency

Processor External Bus Frequency	ROMFAL Value	ROMNAL Value	8-Bit Access Times (in number of clocks)	64-Bit Access Times (in number of clocks)
66.66MHz	10	0	14	98

DRAM

The MPC105 supports one or two banks of DRAM. The PowerBase base board contains one bank (four devices) of 1Mx16 DRAM devices providing 8MB of DRAM, or as a build option, two banks (eight devices) of 1Mx16 DRAM devices providing 16MB of DRAM.

The DRAM is organized as 64-bits wide.

Parity Protection

Optional parity protection is provided by eight 4Mx1 TSOP devices mounted on the back side of the board. The board may be configured with 8MB or 16MB of non-parity memory with only front-side mounted chips, but for parity memory, the back side must also be populated.

Since the parity devices are 4M deep and the two main memory banks are only 1M deep, one 4Mx1 device provides parity for a given byte of both memory banks. Thus, only a total of eight parity devices are required.

A status bit available from the general purpose I/O port allows the firmware to know whether parity is present or not. This bit is controlled by a pulldown resistor.

DRAM Specifications

The PowerBase uses 1Mx16 devices in 400 mil, 50 pin TSOP packaging. The following table lists the DRAM specifications:

Table 3-4. DRAM Specifications

Options		DRAM Specifications			
		Quantity	Size and Configuration	Package	Speed
8MB	Main	4	1M x 16	400 mil TSOP	60ns
16MB	Main	8	1M x 16	400 mil TSOP	60ns

DRAM Timing Configurations

Onboard DRAM devices on the PowerBase are controlled by the MPC105. Refer to the MPC105 specification for additional performance information.

The following table shows the programming values including latency for the MPC105 timing configurations required for 60ns DRAM at the processor/memory bus frequency:

Table 3-5. Programming Values for DRAM Timing Configurations

Processor External Bus Frequency	Read/Write Timing Configurations						DRAM Latency
	RP1	RCD2	CAS3	CP4	CAS5	RAS6P	
66.66MHz	3	2	4	1	3	4	8-4-4-4

Clocks

There are two oscillators on the PowerBase board:

1. A 14.31818MHz oscillator which feeds an MPC980 clock driver chip. The MPC980 provides the 66.6MHz and 33.3MHz clocks for the MPC603 processor, the MPC105

bridge/memory controller, the 82378ZB ISA bridge controller, the VME2PCI chip, and the two PMC slots.

The MPC980 provides the 16MHz clock that is used for the VMEbus SYSCLK when the PowerBase board is configured as the VMEbus system controller.

The MPC980 also provides the 14.31818MHz clock for the counter/timers inside the 82378ZB ISA bridge controller (IBC). .

2. A 1.8432MHz oscillator which is used by the baud rate generator inside the PC16550 UART chip.

VMEbus Interface

The VMEbus interface is provided by the VMEchip2 ASIC. Since the VMEchip2 local bus interface is the MC68040 bus, the VME2PCI ASIC is also required.

The VMEchip2 ASIC, in tandem with the VME2PCI ASIC, constitutes the VMEbus interface. The VMEchip2 interfaces an MC68040-style local bus to the VMEbus. The VME2PCI interfaces the PCI bus to an MC68040-style local bus. When the VMEchip2 and the VME2PCI chips are used together, they form a PCI-bus-to-VMEbus interface.

The VMEchip2/VME2PCI combination provides:

- The local-bus-to-VMEbus interface
- The VMEbus-to-local-bus interface
- The DMA controller functions of the local VMEbus

The VMEchip2 includes Global Control and Status Registers (GCSRs) for interprocessor communications. It can provide the VMEbus system controller functions as well. For detailed programming information, refer to the VMEchip2 and VME2PCI discussions in the *PowerBase Embedded Controller Programmer's Reference Guide*.

VMEchip2 ASIC

The VMEchip2 ASIC is a 324-pin LGA device. Addresses from the VMEbus must be translated by the VMEchip2 to the upper 2GB area since that is where the onboard DRAM is mapped in the PCI memory space. Refer to the *PowerBase Embedded Controller Programmer's Reference Guide* for programming information on the VMEchip2.

VME2PCI ASIC

The VME2PCI ASIC is a 225-pin OMPAC device that interfaces between the PCI Local Bus and the MC68040 bus (the local bus of the VMEchip2). The VME2PCI performs address translation from PCI memory space so that the MPC603 processor can get to the VMEchip2 internal registers, the VMEbus F-page, the VMEbus Short I/O area, and to perform pseudo IACK cycles to fetch interrupt vectors from the VMEchip2 and the VMEbus.

The VME2PCI ASIC also performs byte swapping between PCI and the VMEchip2 since PCI is little-endian and VMEbus is big-endian. Little-endian software may have to manipulate multi-byte data when communicating to the VMEbus.

AD13 is routed to the IDSEL pin on the VME2PCI chip; therefore the base address of the VME2PCI Configuration Space is at $00802000 in the PCI Configuration area. Refer to the *PowerBase Embedded Controller Programmer's Reference Guide* for additional information.

ISA Bridge Controller (IBC)

The PowerBase board uses the Intel 82378ZB to interface to the ISA bus for the debug port, the EEPROM, the Control and Status Registers (CSRs), and the general purpose I/O port. The 82378ZB device, hereafter referred to as the IBC, provides the following features:

- PCI bus arbitration for: MPC105, VME2PCI ASIC, and the two PMC slots

- ISA bus arbitration
- ISA interrupt mapping for four PCI interrupts
- Functionality of two 82C59 interrupt controllers to support 14 ISA interrupts
- Edge/level control for ISA interrupts
- One 16-bit timer
- Three interval counters/timers (82C54 functionality)

AD11 is routed to the IDSEL pin on the IBC device; therefore the base address of the configuration space for the IBC is at $00800800 in the PCI configuration area.

PC16550 UART

The PowerBase board uses a PC16550 Universal Asynchronous Receiver/Transmitter (UART) to provide the asynchronous debug port. TTL-level signals for the port are routed through appropriate EIA-232-D drivers and receivers to an RJ45 connector on the front panel. The external signals are ESD protected.

Timers

Timers and counters on the PowerBase board are provided by the IBC and the VMEchip2.

Interval Timers

The IBC has three built-in counters that are equivalent to those found in an 82C54 programmable interval timer. These counters are grouped into one timer unit, Timer 1, in the IBC. These counters use the OSC clock input as their clock source. The PowerBase drives the OSC pin with a 14.31818MHz clock source.

16-Bit Timers

There is one 16-bit timer provided by the IBC. Refer to the 82378ZB data sheet for programming information on this timer.

VMEchip2 Timers

There are two programmable 32-bit timers in the VMEchip2. Refer to the *PowerBase Embedded Controller Programmer's Reference Guide* for programming information on the VMEchip2.

EEPROM

There is one 28C64 EEPROM device that provides 8KB of non-volatile storage for configuration information and environment variables. Refer to Chapter 6 for more about the configurable environment parameters.

PROM Mezzanine Card Slot

The PowerBase provides for an add-on PROM mezzanine board. The PROM mezzanine interface to the PowerBase is via two 64-pin connectors. These connectors are the same type as those used for PMC boards. The PROM mezzanine provides eight 32-pin PLCC sockets that support one bank of 512Kx8 devices for a total of 4MB of PROM memory. The memory is organized as 64 bits wide and is controlled by the MPC105.

When installed, the PROM mezzanine will disable the onboard PROM/Flash memory because the MPC105 cannot support both memory interfaces simultaneously. The supported package is for one-time programmable (OTP) PLCC devices. The appropriate PROM device type is AM27C040-150JC (or equivalent). Since the devices are socketed, it is assumed that they must be programmed externally before installation on the board.

Note The PROM mezzanine can optionally be configured to use PLCC Flash devices by changing several zero-ohm resistors on the PROM mezzanine board. The Flash devices in such a configuration will not be reprogrammable in-circuit, however. They must be reprogrammed with an external programmer. The appropriate Flash device type is AM29F040-150JC (or equivalent).

The PROM must be accessed in the MPC105 non-burst mode. For PROM speed of 150ns, software should program ROMFAL and ROMNAL in the MPC105 device with the following values:

Table 3-6. Minimum ROMFAL and ROMNAL Values at 150ns

Processor External Bus Speed	Minimum ROMFAL	Minimum ROMNAL
66.66MHz	10	0

A status bit is available from the general purpose I/O port to allow the firmware to know whether it is operating from on-board PROM/Flash or from the PROM mezzanine.

PCI Mezzanine Card (PMC) Slots

The PowerBase board supports two PMC slots. Five 64-pin connectors are located on the PowerBase board to interface to two 32-bit IEEE P1386.1 PMCs to add any desirable function. The PMC slots have the characteristics listed below. For detailed programming information, refer to the programmer's reference guide and to the user documentation for the PMC modules you intend to use.

PMC1

PMC slot 1 supports:

Mezzanine type:	PCI Mezzanine Card (PMC)
Mezzanine size:	Single width and standard depth (75mm x 150mm) with front panel
PMC connectors:	J11, J12, and J14 (32-Bit PCI with front-panel or P2 I/O)
Signalling voltage:	V_{io} = 5.0V

AD16 is routed to the IDSEL pin on PMC slot 1; therefore the base address of the configuration space of this PMC slot is at $00810000 in the PCI configuration area.

PMC2

PMC slot 2 supports:

Mezzanine type:	PCI Mezzanine Card (PMC)
Mezzanine size:	Single width and standard depth (75mm x 150mm) with front panel
PMC connectors:	J21 and J22 (32-Bit PCI with front-panel I/O only)
Signalling voltage:	V_{io} = 5.0V

AD17 is routed to the IDSEL pin on PMC slot 2; therefore the base address of the configuration space of this PMC slot is at $00820000 in the PCI configuration area.

Double-Width PMC

The PMC connectors are located such that a double width PMC may be installed in place of the two single width PMCs.

In this case, the PowerBase supports the following:

Mezzanine type:	PCI Mezzanine Card (PMC)
Mezzanine size:	Double width and standard depth (150mm x 150mm) with front panel
PMC connectors:	J11, J12, J21, J22, and J14 (32-bit PCI with front-panel or P2 I/O)
Signalling voltage:	V_{io} = 5.0V

For the double width PMC, it may pick up AD16 from the PMC slot 1 connector or AD17 from the PMC slot 2 connector; therefore the base address of the configuration space of this PMC may be at $00810000 or $00820000 in the PCI Configuration area.

Transition Board Power

The PowerBase board supplies +12V, -12V, +5V and +3.3V to the PMCs. Because PMC1 is wired directly to the 64 I/O pins of P2, it must provide power to any transition board with which it

communicates. On the PowerBase board, the four pins that are used to provide power to the MVME762 transition module have polyswitches in series for overcurrent protection (factory configuration). A build option, for use with other transition boards, can provide direct connections with no polyswitches.

Note If the pins are used for high-speed signals, you should evaluate the signal integrity.

If power fault protection is desired for transition boards other than the MVME762, it must be provided by the corresponding PMC module since it directly provides power to the transition module.

For the purposes of calculating how many power pins are required for a transition board, the following trace widths are present on PowerBase for the PMC1 I/O pins:

Header	Pin	Trace Width
J14	10	50 mil
	17	75 mil
	28	50 mil
	37	20 mil
	46	20 mil
	55	60 mil
	64	50 mil
	All others	10 mil

Programming the PowerBase 4

Introduction

This chapter provides basic information useful in programming your PowerBase. This includes a description of memory maps, control and status registers, PCI arbitration, interrupt handling, sources of reset, and big/little endian issues.

For complete programming information, refer to the *PowerBase Embedded Controller Programmer's Reference Guide*.

Memory Maps

There are multiple buses on the PowerBase and each bus domain has its own view of the memory map. The following sections describe the PowerBase memory organization from the following three points of view:

- The mapping of all resources as viewed by the processor (MPU bus memory map)
- The mapping of onboard resources as viewed by PCI local bus masters (PCI bus memory map)
- The mapping of onboard resources as viewed by VMEbus masters (VMEbus memory map)

Additional, more detailed memory maps can be found in the *PowerBase Embedded Controller Programmer's Reference Guide*.

MPU Bus Memory Map

The MPU bus memory map is split into different address spaces by the Transfer Type (TT) signals. The local resources respond to the normal access and interrupt acknowledge codes.

Normal Address Range

The memory map of devices that respond to the normal address range is shown in the following tables. The normal address range is defined by the TT signals on the MPU bus. For the MVME130*x*, transfer types 0, 1, and 2 define the normal address range. Table 4-1 defines the entire map ($00000000 to $FFFFFFFF). Many areas of the map are user-programmable, and suggested uses are shown in the table. The cache inhibit function is programmable in the PowerPC 603 microprocessor MMU. The onboard I/O space must be marked “cache inhibit” and serialized in its page table. Table 4-2 focuses on the map for the local I/O devices (accessible through the directly mapped PCI Configuration Space).

Table 4-1. Processor View of the Memory Map

Processor Address		Size	PCI Address Generated		Definition	Notes
Start	End		Start	End		
00000000	7FFFFFFF	2GB			DRAM - Not Forwarded to PCI	
80000000	807FFFFF	8MB	00000000	007FFFFF	ISA/PCI I/O Space	1, 2, 5
80800000	80FFFFFF	8MB	00800000	00FFFFFF	Direct Map PCI Configuration Space	3
81000000	BF7FFFFF	1000MB	01000000	3F7FFFFF	PCI I/O Space	
BF8FFFFF	BFFFFFEF	8MB -16B			Reserved	
BFFFFFF0	BFFFFFFF	16B	3FFFFFF0	3FFFFFFF	PCI IACK/Special Cycles	6
C0000000	C0FFFFFF	16MB	00000000	00FFFFFF	PCI/ISA Memory Space	
C1000000	FEFFFFFF	1GB -32MB	01000000	3EFFFFFF	PCI Memory Space	
Two possible mappings follow, depending on whether the PROM mezzanine is installed. Refer to Note 4.						
FF000000	FFEFFFFF	15MB			Reserved	4
FFF00000	FFF7FFFF	512KB			Onboard PROM/Flash Bank 0	4
FFF80000	FFFFFFFF	512KB			Onboard PROM/Flash Bank 1	4
FF000000	FFBFFFFF	12MB			Reserved	4
FFC00000	FFFFFFFF	4MB			PROM Mezzanine Bank 0	4

Notes

1. PCI configuration accesses to CF8 (Configuration Address) and CFC (Configuration Data) are supported by the MPC105 PCI bridge/memory controller as specified in the PCI Specification Revision 2.0.
2. Both Contiguous and Discontiguous mappings are supported by the PowerBase MVME1300 series. See the *ISA/PCI I/O Space Mapping* section for more details.
3. This space is used for Direct Mapped PCI Configuration Space accesses. See the *PCI Configuration Space Mapping* section for more details.
4. This memory space is mapped as either onboard PROM/Flash space or PROM mezzanine space. (Onboard PROM/Flash memory is disabled when the PROM mezzanine is installed.)

5. The EEPROM is mapped in this area. See the *ISA/PCI I/O Space Mapping* section for more details.
6. A read of any byte within this 16-byte field (BFFFFFF0 through BFFFFFFF) causes a PCI IACK cycle. The data read is the IACK vector.

Direct Mapped PCI Configuration Space

Table 4-2 shows the mapping of the direct mapped PCI configuration space on the PowerBase.

Table 4-2. PCI Configuration Space Map

IDSEL	Processor Address		PCI Configuration Space Address		Definition
	Start	End	Start	End	
	80800000	808007FF	00800000	008007FF	Reserved
A11	80800800	808008FF	00800800	008008FF	IBC Configuration Registers
	80800900	80801FFF	00800900	00801FFF	Reserved
A13	80802000	808020FF	00802000	008020FF	VME2PCI Configuration Registers
	80802100	8080FFFF	00802100	0080FFFF	Reserved
A16	80810000	808100FF	00810000	008100FF	PMC Slot 1 Configuration Registers
	80810100	8081FFFF	00810100	0081FFFF	Reserved
A17	80820000	808200FF	00820000	008200FF	PMC Slot 2 Configuration Registers
	80820100	80FFFFFF	00820100	00FFFFFF	Reserved

Notes

1. Accesses to *Reserved* space may select multiple devices and produce unpredictable results.
2. When a double-width PMC is installed, it may use either A16 or A17 for its IDSEL. During configuration, the firmware will need to probe the PCI bus to find the location of this PMC's configuration space.

ISA/PCI I/O Space

Table 4-3 focuses on the mapping of the ISA/PCI I/O space from the processor view of the memory map.

Table 4-3. ISA/PCI I/O Space Memory Map

ISA I/O Address	Processor Address		Function	Notes
	Contiguous	Discontiguous		
0020 - 0021	8000 0020 - 8000 0021	8000 1000 - 8000 1001	IBC: Interrupt 1 Control & Mask	2
0040 - 0043	8000 0040 - 8000 0043	8000 2000 - 8000 2003	IBC: Timer Counter 1 Registers	2
0060	8000 0060	8000 3000	IBC: Reset Ubus IRQ12	2
0061	8000 0061	8000 3001	IBC: NMI Status and Control	2
0074	8000 0074	8000 3014	EEPROM Address Strobe 0	
0075	8000 0075	8000 3015	EEPROM Address Strobe 1	
0077	8000 0077	8000 3017	EEPROM Data Port	
0092	8000 0092	8000 4012	IBC: Port 92 Register	2
00A0 - 00A1	8000 00A0 - 8000 00A1	8000 5000 - 8000 5001	IBC: Interrupt 2 Control & Mask	2
03F8 - 03FF	8000 03F8 - 8000 03FF	8001 F018 - 8001 F01F	PC16550 UART: Serial Port 1 (DEBUG)	3
04D0	8000 04D0	8002 6010	IBC: INT1 Edge Level Control	2
04D1	8000 04D1	8002 6011	IBC: INT2 Edge Level Control	2
0800	8000 0800	8004 0000	CPU Configuration Register	4
0801	8000 0801	8004 0001	Software Readable Header	4
0802	8000 0802	8004 0002	Board Configuration Register	4
0804	8000 0804	8004 0004	DRAM Size Register	4
0846	8000 0846	8004 2006	General Purpose I/O Register	4

Notes

1. All ISA I/O locations not specified in this table are reserved.
2. These locations are internally decoded by the IBC (PCI/ISA bridge).
3. These locations are internally decoded by the UART.

4. These locations are either not specified by the PRP specification or not PRP-compliant. They may overlap some other functions specified by the PRP specification.
5. The board comes up in contiguous mode. Contiguous and discontiguous modes are programmed by the MPC105 PCI bridge/memory controller.

Caution

The PPCBug debugger and several operating systems execute in contiguous mode. If this is changed to discontiguous mode, PPCBug will cease functioning correctly.

PCI Local Bus Memory Map

Table 4-4 shows the PCI Memory Map of the PowerBase from the point of view of the PCI Local Bus.

Table 4-4. PCI View of the PCI Memory Map

PCI Address		Size	Processor Bus Address		Definition	Notes
Start	End		Start	End		
00000000	00FFFFFF	16MB	Not forwarded to MPU bus		PCI/ISA Memory Space	1, 2
01000000	7FFFFFFF	2GB -16MB	Not forwarded to MPU bus		PCI Memory Space	2
80000000	FFFFFFFF	2GB	00000000	7FFFFFFF	Onboard DRAM (via MPC105)	
00000000	FFFFFFFF	4GB	Not forwarded to MPU bus		PCI/ISA I/O Space	

Notes

1. The IBC (PCI/ISA bridge) performs subtractive decoding in this range and forward the PCI memory cycle to the ISA if DEVSEL_ is not detected.
2. The VME2PCI ASIC can be programmed to claim some of this address range to forward the PCI memory cycle to the VMEchip2.

VMEbus Memory Map

The VMEbus is programmable. The mapping of local resources as viewed by VMEbus masters varies among applications.

The VMEchip2 ASIC includes a user-programmable map decoder for the VMEbus-to-local-bus interface. The map decoder enables you to program the starting and ending address and the modifiers to which the MVME130*x* responds.

The VMEchip2 also includes a user-programmable map decoder for the GCSRs (global control/status registers, accessible from both the VMEbus and the local bus). The GCSR map decoder allows you to program the starting address of the GCSRs in the VMEbus short I/O space.

The VME2PCI ASIC supplies the interface between the PCI local bus and the VMEchip2 ASIC.

Table 4-5 shows the mapping of onboard resources from the point of view of the VME2PCI, and Table 4-6 shows the mapping of onboard resources from the point of view of the VMEchip2.

Table 4-5. VME2PCI View of the Memory Map

Processor Address	PCI Configuration Address	Register Name	Read/Write	Reset Value (Hexadecimal)
80802000	00802000	PCI Vendor ID	R	1057
80802002	00802002	PCI Device ID	R	4800
80802004	00802004	PCI Command	R/W	0000
80802006	00802006	PCI Status	R/W	0000
80802008	00802008	PCI Revision ID	R	01
80802009	00802009	PCI Class Code	R	068000
8080200C	0080200C	PCI Cache Line Size	R/W	00
8080200D	0080200D	PCI Latency Timer	R/W	00
8080200E	0080200E	PCI Header Type	R	00
80802010	00802010	PCI I/O Base Address	R/W	00000001
80802014	00802014	PCI Memory Base Address	R/W	00000000
8080203C	0080203C	PCI Interrupt Line	R/W	00
8080203D	0080200D	PCI Interrupt Pin	R	01
8080203E	0080200E	PCI Minimum Grant	R	00
8080203F	0080200F	PCI Maximum Latency	R	00
80802040	00802040	Slave Starting Address 1	R/W	0000
80802042	00802042	Slave Ending Address 1	R/W	0000
80802044	00802044	Slave Address Offset 1	R/W	0000
80802046	00802046	Slave Address Enable 1	R/W	00
80802048	00802048	Slave Starting Address 2	R/W	0000
8080204A	0080204A	Slave Ending Address 2	R/W	0000
8080204C	0080204C	Slave Address Offset 2	R/W	0000
8080204E	0080204E	Slave Address Enable 2	R/W	00
80802050	00802050	Interrupt Status and Control	R/W	0000

Table 4-6. VMEchip2 Memory Map (Sheet 1 of 3)

VMEchip2 LCSR Base Address = $BASE + 0000
OFFSET:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	SLAVE ENDING ADDRESS 1															
4	SLAVE ENDING ADDRESS 2															
8	SLAVE ADDRESS TRANSLATION ADDRESS 1															
C	SLAVE ADDRESS TRANSLATION ADDRESS 2															
10					ADDER 2	SNP 2			WP 2	SUP 2	USR 2	A32 2	A24 2	BLK D64 2	BLK 2	PRGM 2	DATA2
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
14	MASTER ENDING ADDRESS 1															
18	MASTER ENDING ADDRESS 2															
1C	MASTER ENDING ADDRESS 3															
20	MASTER ENDING ADDRESS 4															
24	MASTER ADDRESS TRANSLATION ADDRESS 4															
28	MAST D16 EN	MAST WP EN	MASTER AM 4						MAST D16 EN	MAST WP EN	MASTER AM 3					
2C	GCSR GROUP SELECT								GCSR BOARD SELECT				MAST 4 EN	MAST 3 EN	MAST 2 EN	MAST 1 EN
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
30											WAIT RMW	ROM ZERO	DMA TB SNP MODE		SRAM SPEED	
34																
38	DMA CONTROLLER															
3C	DMA CONTROLLER															
40	DMA CONTROLLER															
44	DMA CONTROLLER															
48		TICK 2/1	TICK IRQ 1 EN	CLR IRQ	IRQ STAT	VMEBUS INTERRUPT LEVEL			VMEBUS INTERRUPT VECTOR							

This sheet continues on facing page. ⟶

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLAVE STARTING ADDRESS 1															
SLAVE STARTING ADDRESS 2															
SLAVE ADDRESS TRANSLATION SELECT 1															
SLAVE ADDRESS TRANSLATION SELECT 2															
				ADDER 1	SNP 1		WP 1	SUP 1	USR 1	A32 1	A24 1	BLK D64 1	BLK 1	PRGM 1	DATA1
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MASTER STARTING ADDRESS 1															
MASTER STARTING ADDRESS 2															
MASTER STARTING ADDRESS 3															
MASTER STARTING ADDRESS 4															
MASTER ADDRESS TRANSLATION SELECT 4															
MAST D16 EN	MAST WP EN	MASTER AM 2						MAST D16 EN	MAST WP EN	MASTER AM 1					
IO2 EN	IO2 WP EN	IO2 S/U	IO2 P/D	IO1 EN	IO1 D16 EN	IO1 WP EN	IO1 S/U	ROM SIZE		ROM BANK B SPEED			ROM BANK A SPEED		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ARB ROBN	MAST DHB	MAST DWB		MST FAIR	MST RWD	MASTER VMEBUS		DMA HALT	DMA EN	DMA TBL	DMA FAIR	DM RELM		DMA VMEBUS	
DMA TBL INT	DMA LB SNP MODE			DMA INC VME	DMA INC LB	DMA WRT	DMA D16	DMA D64 BLK	DMA BLK	DMA AM 5	DMA AM 4	DMA AM 3	DMA AM 2	DMA AM 1	DMA AM 0
LOCAL BUS ADDRESS COUNTER															
VMEBUS ADDRESS COUNTER															
BYTE COUNTER															
TABLE ADDRESS COUNTER															
DMA TABLE INTERRUPT COUNT				MPU CLR STAT	MPU LBE ERR	MPU LPE ERR	MPU LOB ERR	MPU LTO ERR	DMA LBE ERR	DMA LPE ERR	DMA LOB ERR	DMA LTO ERR	DMA TBL ERR	DMA VME ERR	DMA DONE

1360 9403

← This sheet begins on facing page.

Table 4-6. VMEchip2 Memory Map (Sheet 2 of 3)

VMEchip2 LCSR Base Address = $BASE + 0000
OFFSET:

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
4C								ARB BGTO EN	DMA TIME OFF			DMA TIME ON			VME GLOBAL TIMER	
50	TICK TIMER 1															
54	TICK TIMER 1															
58	TICK TIMER 2															
5C	TICK TIMER 2															
60		SCON	SYS FAIL	BRD FAIL STAT	PURS STAT	CLR PURS STAT	BRD FAIL OUT	RST SW EN	SYS RST	WD CLR TO	WD CLR CNT	WD TO STAT	TO BF EN	WD SRST LRST	WD RST EN	WD EN
64	PRE															
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
68	AC FAIL IRQ	AB IRQ	SYS FAIL IRQ	MWP BERR IRQ	PE IRQ	IRQ1E IRQ	TIC2 IRQ	TIC1 IRQ	VME IACK IRQ	DMA IRQ	SIG3 IRQ	SIG2 IRQ	SIG1 IRQ	SIG0 IRQ	LM1 IRQ	LM0 IRQ
6C	EN IRQ 31	EN IRQ 30	EN IRQ 29	EN IRQ 28	EN IRQ 27	EN IRQ 26	EN IRQ 25	EN IRQ 24	EN IRQ 23	EN IRQ 22	EN IRQ 21	EN IRQ 20	EN IRQ 19	EN IRQ 18	EN IRQ 17	EN IRQ 16
70																
74	CLR IRQ 31	CLR IRQ 30	CLR IRQ 29	CLR IRQ 28	CLR IRQ 27	CLR IRQ 26	CLR IRQ 25	CLR IRQ 24	CLR IRQ 23	CLR IRQ 22	CLR IRQ 21	CLR IRQ 20	CLR IRQ 19	CLR IRQ 18	CLR IRQ 17	CLR IRQ 16
78		AC FAIL IRQ LEVEL				ABORT IRQ LEVEL				SYS FAIL IRQ LEVEL				MST WP ERROR IRQ LEVEL		
7C		VME IACK IRQ LEVEL				DMA IRQ LEVEL				SIG 3 IRQ LEVEL				SIG 2 IRQ LEVEL		
80		SW7 IRQ LEVEL				SW6 IRQ LEVEL				SW5 IRQ LEVEL				SW4 IRQ LEVEL		
84		SPARE IRQ LEVEL				VME IRQ 7 IRQ LEVEL				VME IRQ 6 IRQ LEVEL				VME IRQ 5 IRQ LEVEL		
88	VECTOR BASE REGISTER 0				VECTOR BASE REGISTER 1				MST IRQ EN	SYS FAIL LEVEL	AC FAIL LEVEL	ABORT LEVEL	GPIOEN			
8C																

This sheet continues on facing page. ⟶

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VME ACCESS TIMER		LOCAL BUS TIMER		WD TIME OUT SELECT				PRESCALER CLOCK ADJUST							
COMPARE REGISTER															
COUNTER															
COMPARE REGISTER															
COUNTER															
OVERFLOW COUNTER 2					CLR OVF 2	COC EN 2	TIC EN 2	OVERFLOW COUNTER 1					CLR OVF 1	COC EN 1	TIC EN 1
SCALER															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SW7 IRQ	SW6 IRQ	SW5 IRQ	SW4 IRQ	SW3 IRQ	SW2 IRQ	SW1 IRQ	SW0 IRQ	SPARE	VME IRQ7	VME IRQ6	VME IRQ5	VME IRQ4	VME IRQ3	VME IRQ2	VME IRQ1
EN IRQ 15	EN IRQ 14	EN IRQ 13	EN IRQ 12	EN IRQ 11	EN IRQ 10	EN IRQ 9	EN IRQ 8	EN IRQ 7	EN IRQ 6	EN IRQ 5	EN IRQ 4	EN IRQ 3	EN IRQ 2	EN IRQ 1	EN IRQ 0
SET IRQ 15	SET IRQ 14	SET IRQ 13	SET IRQ 12	SET IRQ 11	SET IRQ 10	SET IRQ 9	SET IRQ 8								
CLR IRQ 15	CLR IRQ 14	CLR IRQ 13	CLR IRQ 12	CLR IRQ 11	CLR IRQ 10	CLR IRQ 9	CLR IRQ 8								
	P ERROR IRQ LEVEL				IRQ1E IRQ LEVEL				TIC TIMER 2 IRQ LEVEL				TIC TIMER 1 IRQ LEVEL		
	SIG 1 IRQ LEVEL				SIG 0 IRQ LEVEL				LM 1 IRQ LEVEL				LM 0 IRQ LEVEL		
	SW3 IRQ LEVEL				SW2 IRQ LEVEL				SW1 IRQ LEVEL				SW0 IRQ LEVEL		
	VME IRQ 4 IRQ LEVEL				VMEB IRQ 3 IRQ LEVEL				VME IRQ 2 IRQ LEVEL				VME IRQ 1 IRQ LEVEL		
GPIOO				GPIOI				GPI							
								MP IRQ EN	REV EROM	DIS SRAM	DIS MST	NO EL BBSY	DIS BSYT	EN INT	DIS BGN

1361 9403

← This sheet begins on facing page.

Table 4-6. VMEchip2 Memory Map (Sheet 3 of 3)

VMEchip2 GCSR Base Address = $BASE + 0100

Offsets																	
VME-bus	**Local Bus**	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CHIP REVISION								CHIP ID							
2	4	LM3	LM2	LM1	LM0	SIG3	SIG2	SIG1	SIG0	RST	ISF	BF	SCON	SYSFL	X	X	X
4	8	GENERAL PURPOSE CONTROL AND STATUS REGISTER 0															
6	C	GENERAL PURPOSE CONTROL AND STATUS REGISTER 1															
8	10	GENERAL PURPOSE CONTROL AND STATUS REGISTER 2															
A	14	GENERAL PURPOSE CONTROL AND STATUS REGISTER 3															
C	18	GENERAL PURPOSE CONTROL AND STATUS REGISTER 4															
E	1C	GENERAL PURPOSE CONTROL AND STATUS REGISTER 5															

Note Refer to the *PowerBase Embedded Controller Programmer's Reference Guide* for the bit and register functions of the last six registers above when PPC1Bug uses the registers.

VME2PCI ASIC Programming Model

The VME2PCI ASIC provides the necessary interface between the PCI Local Bus and the VMEchip2.

The Control and Status Registers of the VME2PCI ASIC consist of: The VMEchip2 CSR window, the Pseudo IACK Registers, and the PCI Configuration Registers. All these registers can be mapped anywhere in the PCI Memory Space and/or PCI I/O Space. The mapping is done by programming the appropriate base address registers of the VME2PCI Configuration Registers. In addition, the PCI Configuration Registers are also accessible from the PCI Configuration Space. Figures 4-1 and 4-2 illustrate how the VME2PCI Control and Status Registers can be mapped:

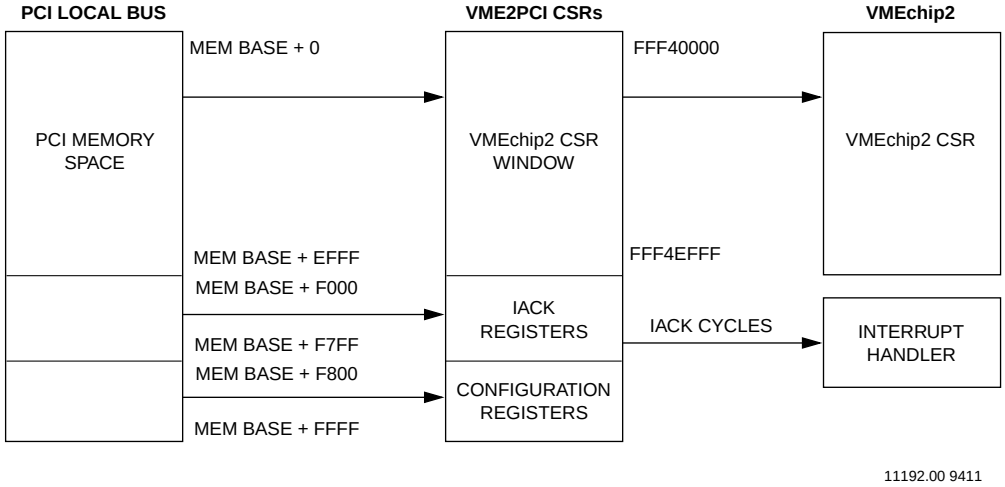


Figure 4-1. VME2PCI's CSR Mapping in PCI Memory Space

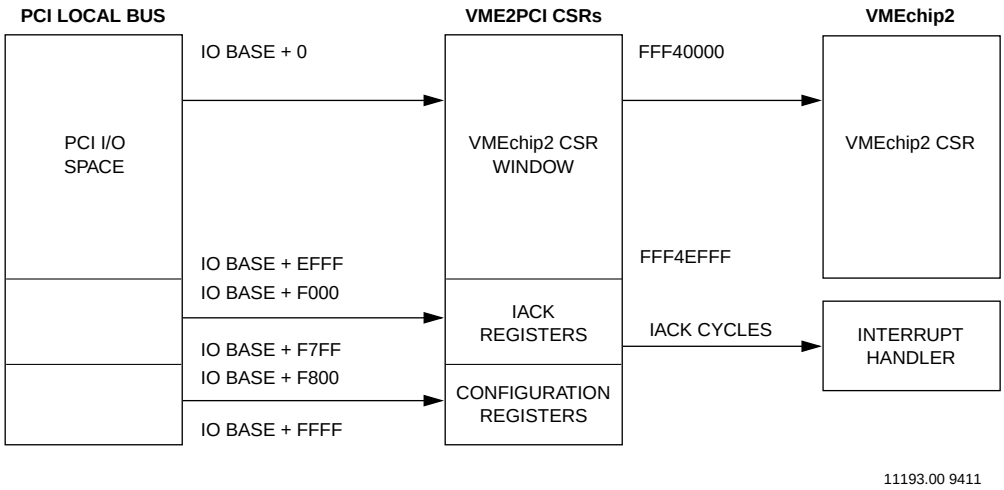


Figure 4-2. VME2PCI's CSR Mapping in PCI I/O Space

EEPROM Memory Map

The EEPROM is divided into five areas as shown in Table 4-7. The first four areas are defined by software, while the fifth area is reserved for future use. The first area is reserved for user data. The second area is open. The third area is used by the PowerBase board debugger (PPCBug). The fourth area, detailed in Table 4-8, is the configuration area.

The EEPROM chip is not a direct address mapped device. The EEPROM address strobe registers, in conjunction with the EEPROM data port register, must be used to gain access (i.e., read/write of data).

Table 4-7. EEPROM Memory Map

Offset (start/end)	Description	Size (Bytes)
$00000000 - $00000FFF	EEPROM per the PRP specification	4096
$00001000 - $000010FF	Open	1784
$000016F8 - $00001EF7	Debugger area	2048
$00001EF8 - $00001FF7	Configuration area (see Table 4-8)	256
$00001FF8 - $00001FFF	Reserved for future use	8

Table 4-8. EEPROM Configuration Area Memory Map

Offset (start/end)	Description	Size (Bytes)
$00001EF8 - $00001EFB	Version	4
$00001EFC - $00001F07	Serial number	12
$00001F08 - $00001F17	Board ID	16
$00001F18 - $00001F27	PWA	16
$00001F28 - $00001F2B	Reserved_0	4
$00001F2C - $00001F31	Ethernet address	6
$00001F32 - $00001F33	Reserved_1	2
$00001F34 - $00001F35	Local SCSI ID	2
$00001F36 - $00001F38	MPU Speed in MHz	3
$00001F39 - $00001F3B	Bus Speed in MHz	3
$00001F3C - $00001FF6	Reserved	187
$00001FF7	Checksum	1

Control and Status Registers

The PowerBase has the following Control and Status Registers (CSRs):

- CPU Configuration Register
- Software Readable Header Register
- Board Configuration Register
- DRAM Size Register
- General Purpose I/O Port

Programming information for these registers is provided in the *PowerBase Embedded Controller Programmer's Reference Guide.*

CPU Configuration Register

The CPU Configuration Register provides configuration information about the processor.

REG	CPU Configuration Register -$0800							
BIT	SD7	SD6	SD5	SD4	SD3	DS2	SD1	SD0
FIELD	CPUTYPE				CKM1	CKM0	L2P1	L2P0
OPER	R				R	R	R	R
RESET	0011b				1	1	1	1

L2P1-L2P0 L2 Cache present. These bits are defined as follows:

L2P1	L2P0	L2 Cache Size
0	0	512KB
0	1	256KB
1	0	1MB
1	1	L2 Cache Not Present

For the PowerBase, this field is hardwired to 11b.

CKM1-CKM0 Clocking configuration. These bits reflect the clocking configuration of the PowerBase. The encoding for these bits is as follows:

CKM1	CKM0	PCI Bus Clock	CPU External Bus Clock
0	0	33MHz	33MHz
0	1	20MHz	40MHz
1	0	25MHz	50MHz
1	1	33.33MHz	66.66MHz

For the PowerBase, this field is hardwired to 11b.

CPUTYPE CPU type. These four bits reflect the CPU type information.

For the PowerBase, this field is hardwired to 11b.

Software Readable Header Register

The Software Readable Header, J2 on the PowerBase, controls a read-only register located at ISA I/O address $x801. A jumper installed for a particular bit results in a logic 0, and no jumper results in a logic 1.

With the jumper installed between pins 3 and 4 (factory configuration), the debugger uses the current user setup/operation parameters in EEPROM. When the jumper is removed (making the bit a **1**), the debugger uses the default setup/operation parameters in ROM instead. Refer to the **ENV** command description in Chapter 6 for the ROM defaults.

The five higher-order bits, SRH3 to SRH7, are required to get GCSR locations for up to 18 boards when used in a Wide Area Network (WAN).

Refer also to Chapter 1, *Setting the General-Purpose Software Header (J2).*

Table 4-9. Software Readable Header Jumpers

J2 Pins	SRH Bit	Definition
1 and 2	Bit #0 (SRH0)	Reserved for future use.
3 and 4	Bit #1 (SRH1)	Used for setup/operation parameters selection.
5 and 6	Bit #2 (SRH2)	Reserved for future use.
7 and 8	Bit #3 (SRH3)	Used for board selection in WAN.
9 and 10	Bit #4 (SRH4)	Used for board selection in WAN.
11 and 12	Bit #5 (SRH5)	Used for board selection in WAN.
13 and 14	Bit #6 (SRH6)	Used for board selection in WAN.
15 and 16	Bit #7 (SRH7)	Used for board selection in WAN.

Board Configuration Register

The Board Configuration Register is an 8-bit register providing the configuration information about the PowerBase.

REG	Board Configuration Register - 0802 (hex)							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	N/A	SCCP_	PMC2P_	PMC1P_	VMEP_	GFXP_	LANP_	SCSIP_
OPER	R	R	R	R	R	R	R	R
RESET	1	1	N/A	N/A	0	1	1	1

(Not Used) Always set. (Formerly known as Transition Module Present.) The PowerBase boards have no way of knowing if a transition module is present.

SCCP_ Always set. Z85230 ESCC Present. If set, there is no on-board sync serial support (ESCC not present). If cleared, there is on-board support for sync serial interface via Z85230 ESCC. (Note that this pertains to an ESCC mounted on the base board, not on a PMC.)

PMC1P_ PMC1 Present. If set, there is no PCI Mezzanine Card installed in PMC Slot 1. If cleared, PMC slot 1 contains a PMC.

PMC2P_ PMC2 Present. If set, there is no PCI Mezzanine Card installed in PMC Slot 2. If cleared, PMC slot 2 contains a PMC.

Note A double-width PMC may use either PMC1P_ or PMC2P_ as its presence detect bit.

VMEP_ Always cleared. VMEbus Present. If set, there is no VMEbus interface. If cleared, VMEbus interface is supported.

GFXP_ Always set. Graphics Present. If set, there is no onboard graphics interface. If cleared, there is an onboard graphics capability (MVME1300 series has no graphics).

LANP_ Always set. Ethernet Present. If set, there is no Ethernet transceiver interface. If cleared, there is onboard Ethernet support (MVME1300 series has no onboard Ethernet support).

SCSIP_ Always set. SCSI Present. If set, there is no onboard SCSI interface. If cleared, onboard SCSI is supported (MVME1300 series has no onboard SCSI interface).

DRAM Size Register

The DRAM Size Register is an 8-bit register providing the DRAM size information. Banks 0 (and 1, if present) are on the PowerBase s board.

REG	DRAM Size Register - 0804h							
BIT	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
FIELD	B2/B3 ASYM_	B2/B3 SIZ2	B2/B3 SIZ1	B2/B3 SIZ0	B0/B1 ASYM_	B0/B1 SIZ2	B0/B1 SIZ1	B0/B1 SIZ0
OPER	R	R	R	R	R	R	R	R
RESET	N/A	1	1	1	N/A	N/A	N/A	N/A

SIZ2-SIZ0 DRAM Size. These bits provide the DRAM size information for the two banks of DRAM supported by the PowerBase. The encoding for these size bits is as follows:

B0/B1 (B2/B3)			DRAM Size	
SIZ2	SIZ1	SIZ0	Bank 0 (Bank 2)	Bank 1 (Bank 3)
0	1	1	Not Present	Not Present
0	1	0	8MB	Not Present
0	0	1	32MB	Not Present

0	0	0	128MB	Not Present
1	1	1	Not Present	Not Present
1	1	0	8MB	8MB
1	0	1	32MB	32MB
1	0	0	128MB	128MB

Note The only valid combinations for PowerBase are 010 (binary) and 110 (binary).

ASYM_ Asymmetric Refresh Mode. When cleared, this bit indicates that the DRAM devices installed for Bank 0 and Bank 1 (Bank 2 and Bank 3) have more row address bits than column address bits. This bit is used to determine how to program the MPC105 chip appropriately. For 1M x16 DRAM, the asymmetric refresh mode is also referred to as the 4K refresh mode. For these devices, there would be 12 row address bits and 8 column address bits.

General Purpose I/O Port

The general purpose I/O port is used to provide various functions. This port is mapped into the ISA I/O space. The assignments for the port pins are as shown in Table 4-10:

Table 4-10. General Purpose I/O Port Pins Assignment

Port Pin	Signal Name	Direction	Signal Name
PA0	BRDFAIL0	Output	Board Fail: when set will cause FAIL LED to be lit.
PA1	TBENDIS_	Output	TBEN Disable. If cleared, TBENDIS_ will drive processor's TBEN pin low to disable its internal timebase.
PA2	N/A	Output	N/A
PA3	N/A	Output	N/A
PA4	FNR	Input	FNR = 0 means PROMmez is installed.
PA5	PARITY_	Input	PARITY_ = 0 means parity DRAM is present.

Table 4-10. General Purpose I/O Port Pins Assignment (Continued)

Port Pin	Signal Name	Direction	Signal Name
PA6	FUSE	Input	FUSE = 1 means at least one of the polyswitches on the PowerBase MVME1300 series board is opened.
PA7	ABORT_	Input	Status of the ABORT_ signal.

Programming Considerations

Good programming practice dictates that only one MPU at a time have control of the MVME130*x* control registers. Of particular note are:

- Registers that modify the address map
- Registers that require two cycles to access
- VMEbus interrupt request registers

PCI Arbitration

There are five potential PCI bus masters on the MVME130*x* embedded controller:

- MPC105 (PCI/MPU bus bridge and memory controller)
- IBC (PCI/ISA bus bridge controller)
- VME2PCI ASIC (PCI/VMEchip2 interface ASIC)
- Two PMC (PCI mezzanine card) slots

The IBC supplies the PCI arbitration support for these five devices. The IBC supports flexible arbitration modes of fixed priority, rotating priority, and mixed priority.

The IBC registers that control the arbitration mode are the PCI Arbiter Priority Control (PAPC) Register and the PCI Arbiter Priority Control Extension (ARBPRIX) Register. The PAPC register and the ARBPRIX register default to 04 (hex) and 00 (hex)

respectively. This default configuration puts the CPU (MPC105) at the highest priority level. Refer to the S82378ZB Reference Manual for programming information.

Table 4-12 shows the PCI arbitration assignments for all PCI masters on the PowerBase.

Figure 4-3 shows the arbitration configuration diagram of the IBC.

Additional details on PCI arbitration can be found in the *PowerBase Embedded Controller Programmer's Reference Guide*.

Table 4-11. PCI Arbitration Assignments

PCI BUS REQUEST	CPUREQ*	IBCREQ*	REQ0*	REQ1*	REQ2*	REQ3*
PCI MASTER	CPU (MPC105)	IBC (Internal)	PMC1 (Slot 1)		VME (VME2PCI)	PMC2 (Slot 2)

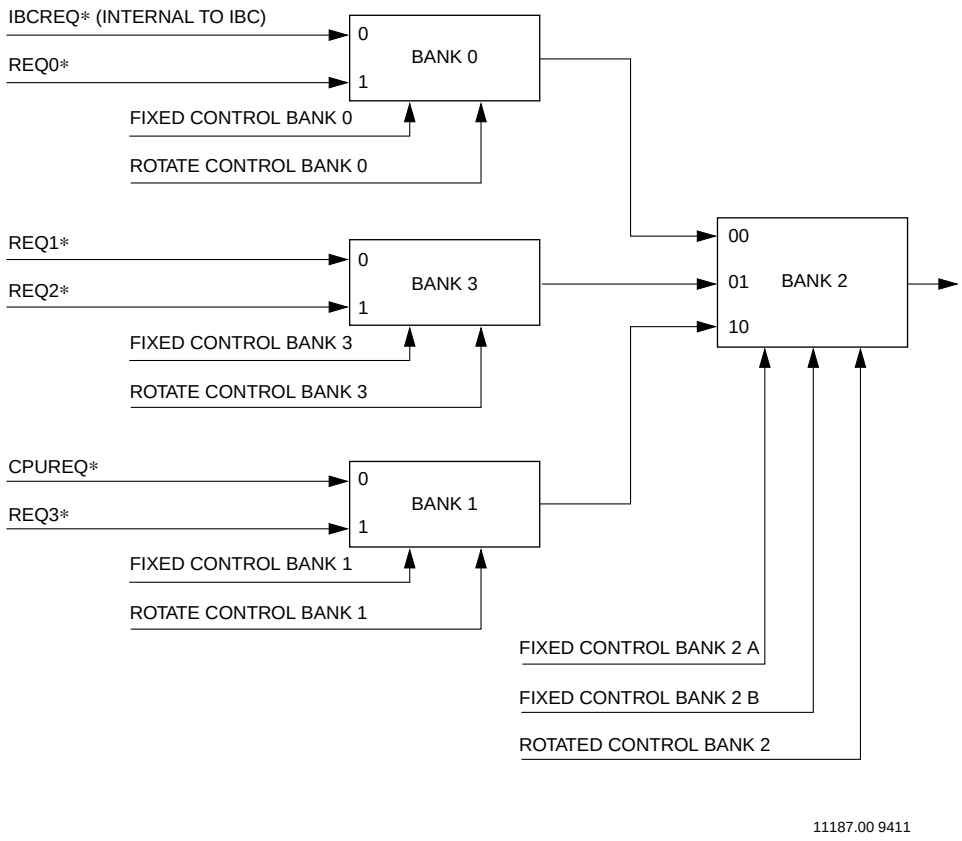


Figure 4-3. IBC Arbiter Configuration Diagram

Interrupt Handling

The MVME130*x* supports both maskable and non-maskable interrupts. Figure 4-4 illustrates the interrupt architecture.

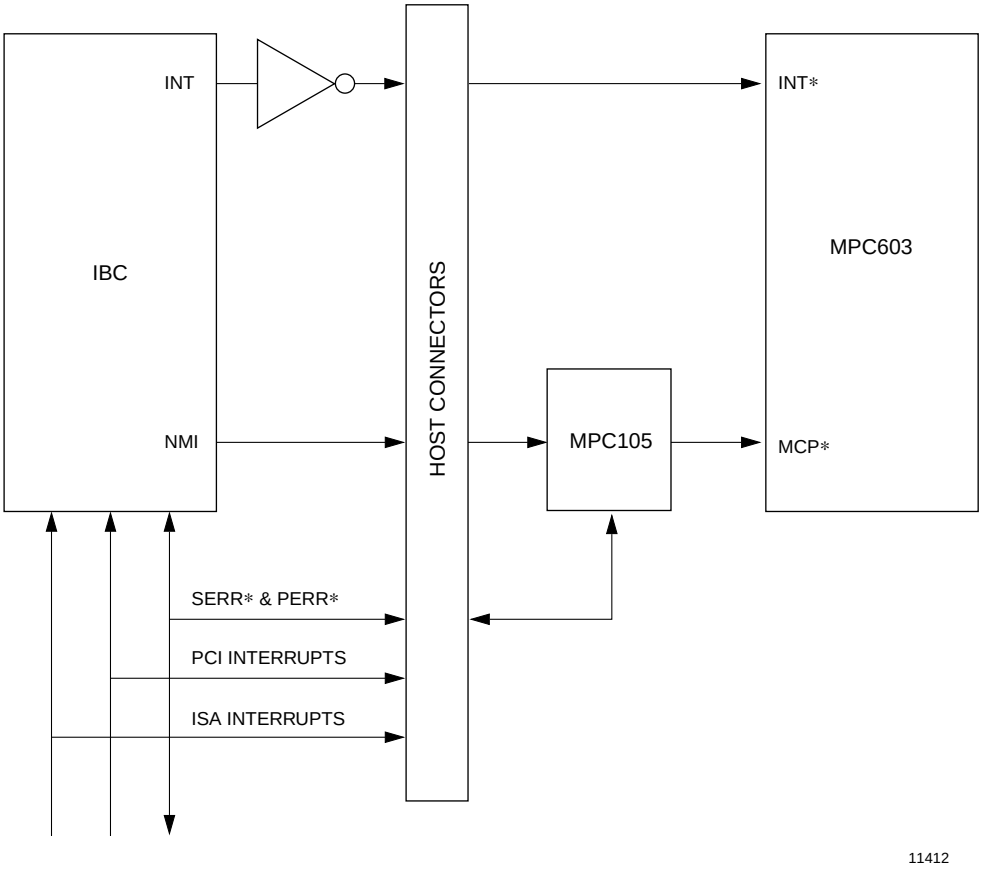


Figure 4-4. MVME130*x* Interrupt Architecture

Machine Check Interrupt (MCP*)

The IBC can be programmed to assert NMI when it detects either SERR* low on the PCI Local Bus or IOCHK* low on the ISA bus. However, IOCHK* is not used on the MVME130x. The MPC105 will assert MCP* to the processor upon detecting a high level on NMI from the IBC.

Note that MPC105 also monitors SERR* and PERR*. It can be programmed to asserted MCP* when it detects a low level on either SERR* or PERR*.

The MPC105 can also be programmed to assert MCP* under many other conditions. Refer to the *PowerBase Embedded Controller Programmer's Reference Guide* for additional information on the MCP* interrupt signal.

Maskable Interrupts

The IBC supports 15 interrupt requests. These 15 interrupts are ISA-type interrupts that are functionally equivalent to two 82C59 interrupt controllers. Except for IRQ0, IRQ1, IRQ2, IRQ8*, and IRQ13, each of the interrupt lines can be configured for either edge-sensitive or level-sensitive mode by programming the appropriate ELCR registers in the IBC.

There IBC also supports four PCI interrupts: INT3*-INT0*. The IBC has four PIRQ Route Control Registers to allow each PCI interrupt line to be routed to any of eleven ISA interrupt lines (IRQ0, IRQ1, IRQ2, IRQ8*, and IRQ13 are reserved for ISA system interrupts). Since PCI interrupts are defined as level-sensitive, software must program the selected IRQ(s) for level-sensitive mode. Note that more than one PCI interrupt can be routed to the same ISA IRQ line.

Figure 4-5 shows the IBC interrupt structure. Additional details on interrupt assignments can be found in the *PowerBase Embedded Controller Programmer's Reference Guide*.

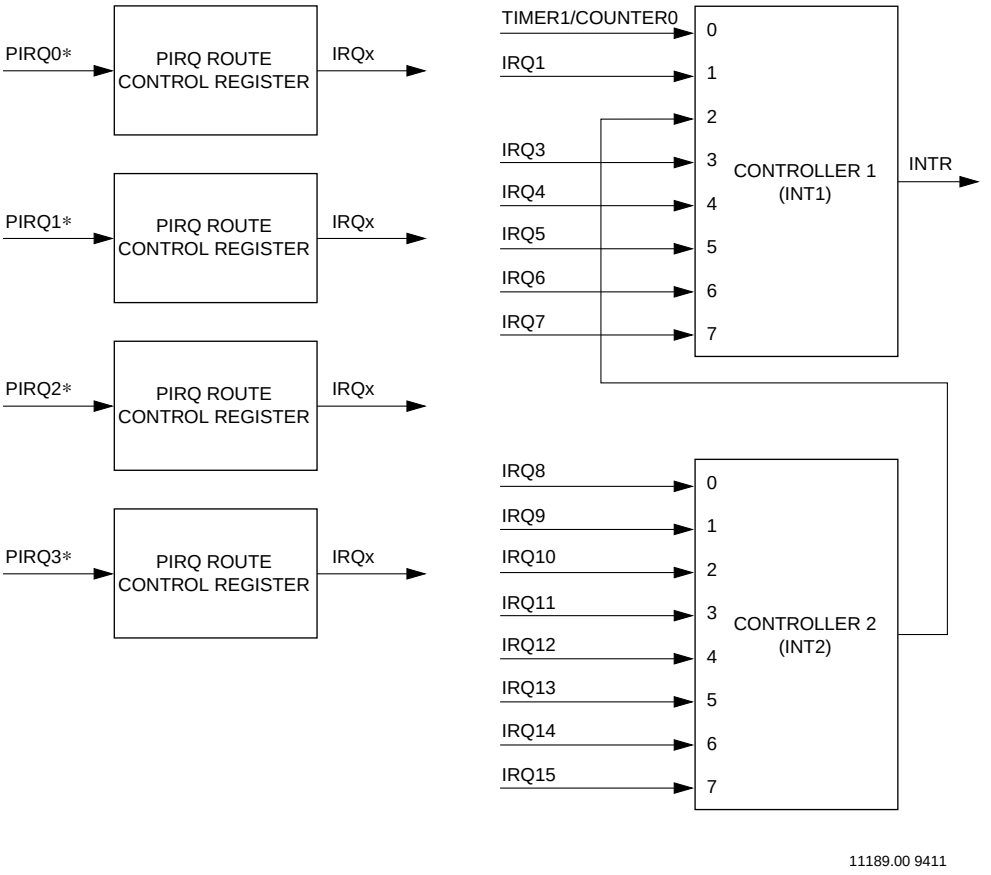


Figure 4-5. IBC Interrupt Handler Block Diagram

Handling VMEchip2 Interrupts

VMEchip2 interrupts consist of interrupts from the VMEbus IRQ lines and from the VMEchip2 internal resources (i.e., DMA and Timers). You can program the VMEchip2 interrupt control registers as though the system were MC68040-based (i.e., with interrupt priority levels from 1 through 7). When an interrupt is pending, the VMEchip2 asserts three encoded interrupt request lines (IPL2*-IPL0*) to the VME2PCI device. An interrupt is then issued by the VME2PCI device to the processor through the IBC.

After learning from the IBC that the source of the interrupt is the VME2PCI, the software determines the interrupt level to acknowledge the VMEchip2 by examining the ILVL status bits of the Interrupt Control and Status Register in the VME2PCI ASIC. Finally, to get the interrupt vector from the VMEchip2, the interrupt handling routine must read the appropriate Pseudo IACK Registers.

Handling ABORT Interrupts

The MVME130*x* can be programmed to generate an interrupt to the processor via ISA Interrupt IRQ8* when the ABORT switch is activated (refer also to the *ABORT Switch* section at the beginning of this chapter). The ABORT* signal is also routed to the general purpose I/O port. Refer to the 82378ZB Data Sheet for programming information.

DMA Channels

The IBC supports seven DMA channels. The PowerBase does not use the DMA channels in the IBC.

Sources of Reset

The MVME130*x* embedded controller has six equally powerful potential sources of reset:

1. Power-on reset.
2. RESET switch.
3. ALT_RST* function controlled by the Port 92 register in the IBC (resets the VMEbus when the MVME130*x* is system controller).
4. Reset sources from the VMEchip2: the VMEbus SYSRESET*, Watchdog Reset, and Software Reset functions.
5. When the MVME130*x* is operating as the VMEbus System Controller, an HRESET* signal will also cause a VMEbus SYSRESET*.

Endian Issues

The PowerBase supports both little-endian (e.g. Windows NT) and big-endian software (e.g. AIX). The PowerPC processor and the VMEbus are inherently big-endian, while the PCI bus is inherently little-endian. The following figures illustrate how the MVME130*x* handles the endian issue in big-endian and little-endian modes:

Processor/Memory Domain

The MPC603 processor can operate in both big-endian and little-endian mode. However, it always treats the external processor/memory bus as big-endian by performing *address rearrangement and reordering* when running in little-endian mode.

Role of the MPC105

Because the PCI bus is little-endian, the MPC105 performs byte swapping in both directions (from PCI to memory and from the processor to PCI) to maintain address invariance while programmed to operate in big-endian mode with the processor and the memory subsystem.

In little-endian mode, the MPC105 *reverse-rearranges* the address for PCI-bound accesses and *rearranges* the address for memory-bound accesses (from PCI). In this case, no byte swapping is done.

PCI Domain

The PCI bus is inherently little-endian and all devices connected directly to PCI will operate in little-endian mode, regardless of the mode of operation in the processor's domain.

Role of the VME2PCI

Because PCI is little-endian and the VMEbus is big-endian, the VME2PCI performs byte swapping in both directions (from PCI to VMEbus and from VMEbus to PCI) to maintain address invariance, regardless of the mode of operation in the processor's domain.

VMEbus Domain

The VMEbus is inherently big-endian. All devices connected directly to the VMEbus are expected to operate in big-endian mode, regardless of the mode of operation in the processor's domain.

In big-endian mode, byte-swapping is performed first by the VME2PCI and then by the MPC105. The result has the desirable effect of being transparent to the big-endian software.

In little-endian mode, however, software must take the byte-swapping effect of the VME2PCI and the address *reverse-rearranging* effect of the MPC105 into account.

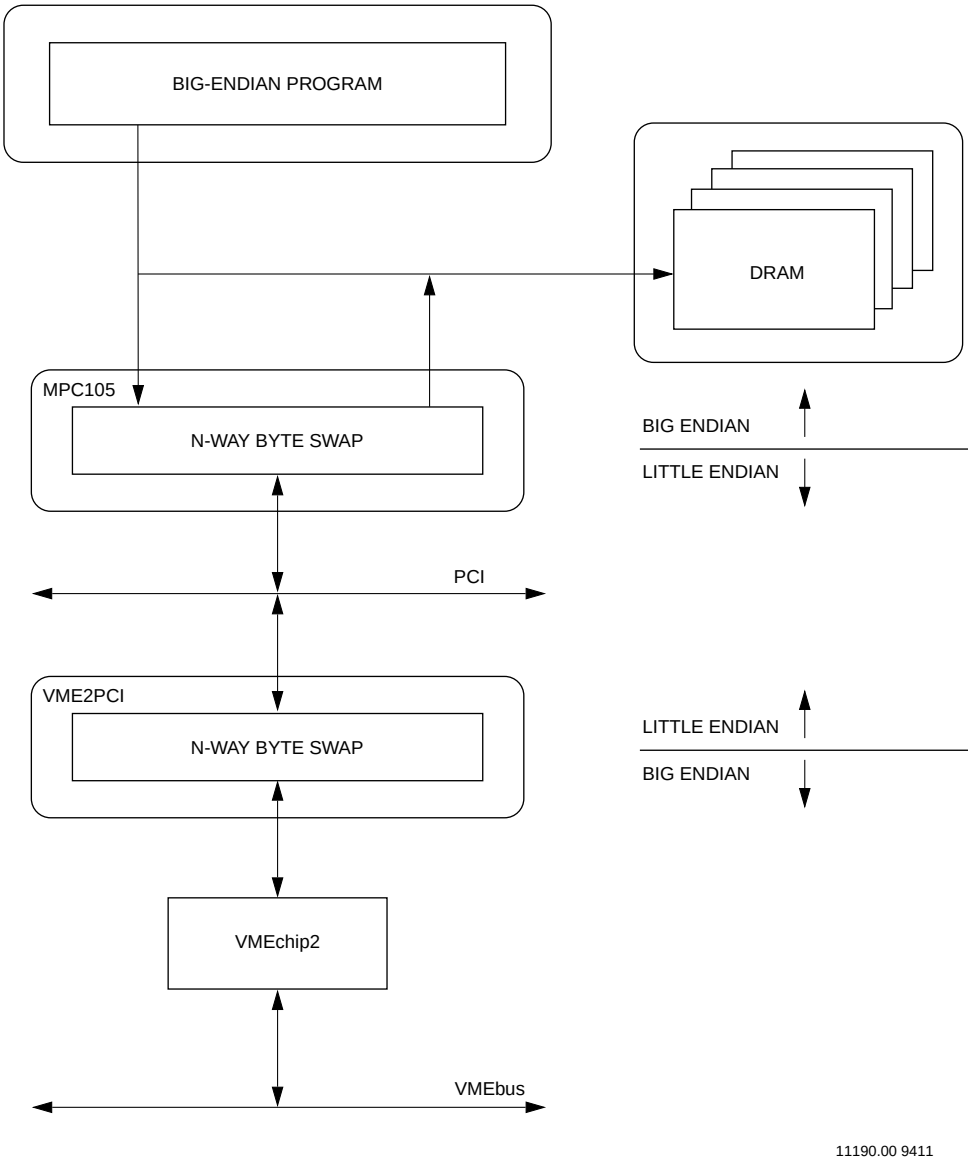


Figure 4-6. Big-Endian Mode

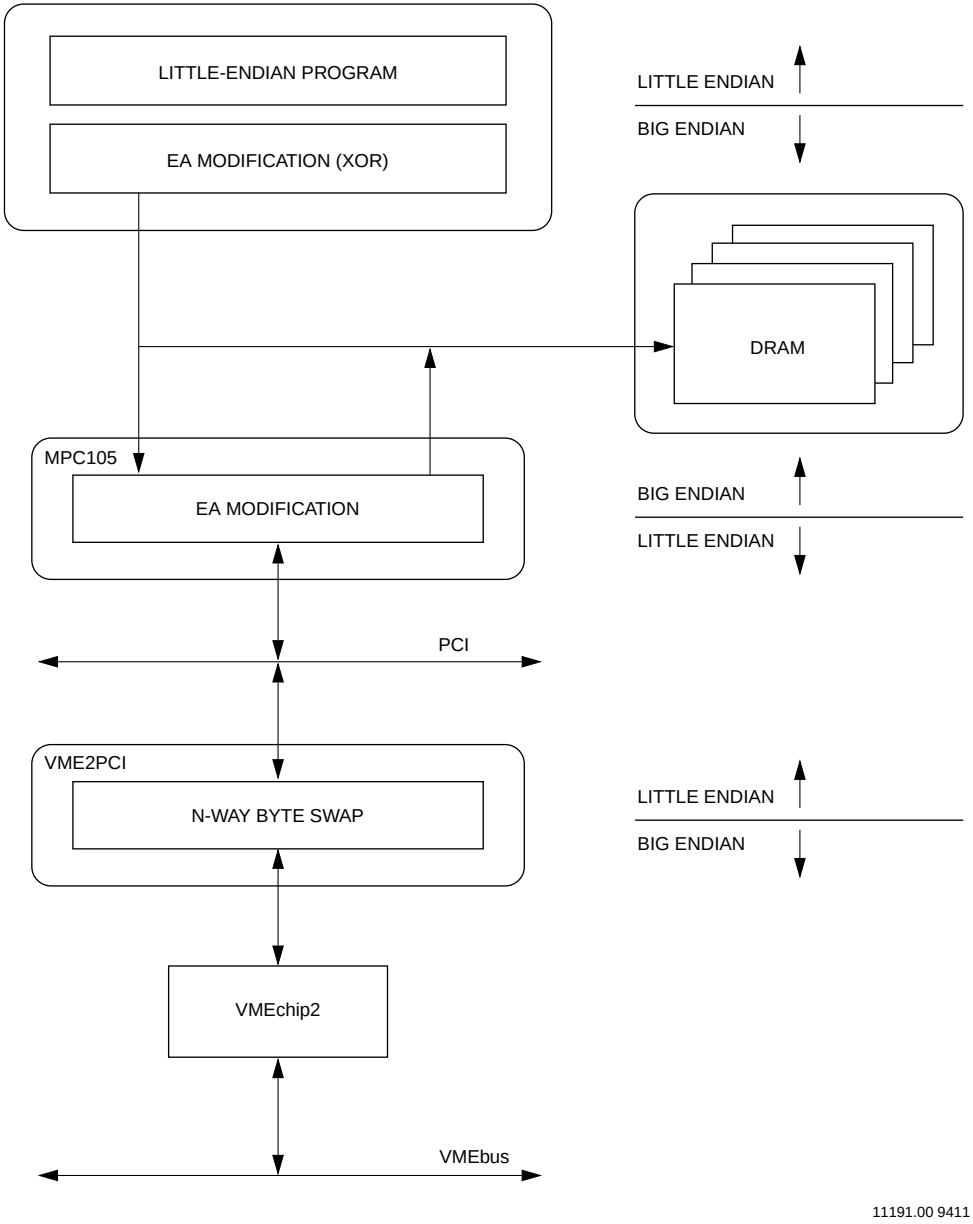


Figure 4-7. Little-Endian Mode

Using PPCBug 5

PPCBug Overview

The PPCBug firmware is the layer of software just above the hardware. The firmware provides the proper initialization for the devices on the PowerBase board upon power-up or reset.

This chapter describes the basics of PPCBug and its architecture, describes the monitor (interactive command portion of the firmware) in detail, and gives information on actually using the PPCBug debugger and the special commands. A complete list of PPCBug commands appears at the end of the chapter.

Chapter 6 contains information about the CNFG and ENV commands, system calls, and other advanced user topics.

PPCBug Basics

The PowerPC debug firmware, PPCBug, is a powerful evaluation and debugging tool for systems built around the Motorola PowerPC microcomputers. Facilities are available for loading and executing user programs under complete operator control for system evaluation.

PPCBug provides a high degree of functionality, user friendliness, portability, and ease of maintenance.

It achieves good portability and comprehensibility because it was written entirely in the C programming language, except where necessary to use assembler functions.

PPCBug includes commands for:

- Display and modification of memory
- Breakpoint and tracing capabilities

- A powerful assembler and disassembler useful for patching programs
- A self-test at power-up feature which verifies the integrity of the system

PPCBug consists of three parts:

- A command-driven, user-interactive *software debugger*, described in the *PPCBug Firmware Package User's Manual*. It is hereafter referred to as "the debugger" or "PPCBug".
- A command-driven *diagnostics package* for the PowerBase hardware, hereafter referred to as "the diagnostics." The diagnostics package is described in the *PPC1Bug Diagnostics Manual.*
- A *user interface* or *debug/diagnostics monitor* that accepts commands from the system console terminal.

When using PPCBug, you operate out of either the *debugger directory* or the *diagnostic directory.*

- If you are in the debugger directory, the debugger prompt `PPC1-Bug>` is displayed and you have all of the debugger commands at your disposal.
- If you are in the diagnostic directory, the diagnostic prompt `PPC1-Diag>` is displayed and you have all of the diagnostic commands at your disposal as well as all of the debugger commands.

Because PPCBug is command-driven, it performs its various operations in response to user commands entered at the keyboard. When you enter a command, PPCBug executes the command and the prompt reappears. However, if you enter a command that causes execution of user target code (e.g., **GO**), then control may or may not return to PPCBug, depending on the outcome of the user program.

PPCBug Implementation

Physically, PPCBug is contained in two Flash devices on the PowerBase that together provide 1MB of storage.

Caution

If the optional PROM mezzanine board is mounted on the PowerBase, the firmware on the PowerBase board is no longer accessible.

PPCBug is written largely in the C programming language, providing benefits of portability and maintainability. Where necessary, assembler has been used in the form of separately compiled program modules containing only assembler code. No mixed language modules are used.

The executable code is checksummed at every power-on or reset firmware entry. The result is checked with a pre-calculated checksum contained in the last 16-bit word of the Flash image.

Comparison with Other MCG Debuggers

PPCBug is similar to previous Motorola Computer Group (MCG) firmware debugging packages such as MVME147Bug, MVME167Bug, and MVME187Bug, with differences due to microprocessor architectures. These differences are primarily reflected in the:

- Instruction mnemonics
- Register displays
- Addressing modes of the assembler/disassembler
- Argument passing to the system calls

Power-Up/Reset Sequence

Figure 5-1 illustrates the basic flow the firmware follows at a power-up/reset sequence.

On the PowerBase, the default power-up condition is to the setup parameters contained in EEPROM. Using the **ENV** command (Chapter 6), you can change this if desired.

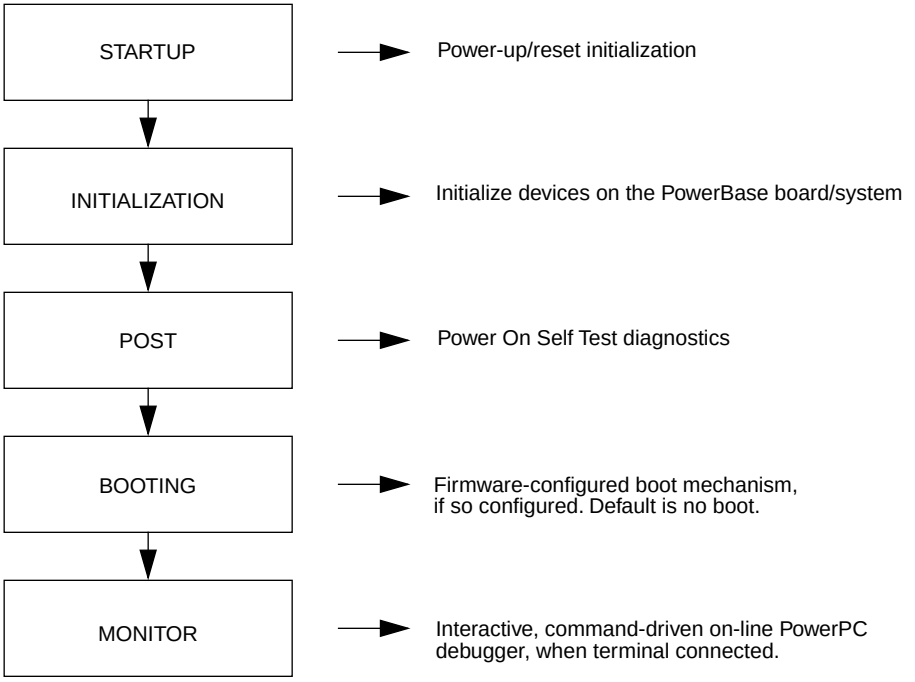


Figure 5-1. PowerPC Debugger Architecture

Debug Monitor

The debug "monitor" (interactive command level) is available if a terminal is connected to the DEBUG port. At the monitor, a number of different commands may be entered to interact with the hardware. Specifically, the commands are routed through various drivers in the firmware. This way the actual register settings and commands used by the hardware are transparent to a firmware user. The firmware user only needs to be familiar with the basic PPCBug commands.

Restarting the PowerBase

You can initialize the PowerBase to a known state in three different ways:

- Break
- Reset
- Abort

Each has certain characteristics which make it more appropriate than the others in given situations.

Break

A "break" is generated by pressing and releasing the BREAK key on the current-console keyboard. Break does not generate an interrupt. The only time break is recognized is when characters are sent or received by the console port. Break performs the following:

- Removes any breakpoints in the user code.
- Keeps the breakpoint table intact.
- Takes a snapshot of the machine state if the function was entered using SYSCALL.
- Allows access to the snapshot for diagnostic purposes.

Many times it may be desirable to terminate a debugger command prior to its completion; for example, the display of a large block of memory. Break allows you to terminate the command immediately.

Reset

A system reset is initiated by pressing and releasing the PowerBase board's RESET switch.

Cold and warm reset modes are available. By default, PPCBug is in cold mode (refer to the **RESET** command description in the *PPCBug Firmware Package User's Manual*). During cold reset, a total system initialization takes place, as if the PowerBase had just been powered up:

- All static variables are restored to their default states.
- The breakpoint table and offset registers are cleared.
- The target registers are invalidated. Input and output character queues are cleared.
- Onboard devices are reset.
- The first two serial ports are reconfigured to their default state.

During warm reset, the PPCBug variables and tables are preserved, as well as the target state registers and breakpoints. Note that revision 1.1 of the PPCBug does not support the warm reset feature.

Reset must be used if the processor ever halts, or if the PPCBug environment is ever lost (vector table is destroyed, stack corrupted, etc.).

Abort

Abort is invoked by pressing and releasing the PowerBase board's ABORT switch.

Whenever abort is invoked when executing a user program (running target code), a "snapshot" of the processor state is captured and stored in the target registers. When working in the debugger, abort captures and stores only the following:

- Instruction pointer
- Status register
- Format and vector information

For this reason, abort is most appropriate when terminating a user program that is being debugged. Abort should be used to regain control if the program gets caught in a loop, etc. The target IP and register contents help to pinpoint the malfunction.

Pressing and releasing the abort switch causes the following:

- An interrupt is sent to the microprocessor.
- The target registers, reflecting the machine state at the time the abort switch was pressed, are displayed on the screen.
- Any breakpoints installed in the user code are removed.
- Breakpoint table remains intact.
- Control is returned to the debugger.

Board Failure

The following conditions result in a board failure:

- Board initialization error/failure
- Debugger object checksum error
- Environment data (EEPROM **ENV** parameters) failure (i.e., checksum)
- Configuration data (EEPROM **CNFG** parameters) failure (i.e., checksum)

- Calculated MPU clock speed does not match the associative **CNFG** parameter
- Calculated BUS clock speed does not match the associative **CNFG** parameter
- Selftest error/failure

MPU Clock Speed Calculation

The MPU clock speed is calculated and checked against a user definable parameter housed in EEPROM (refer to the **CNFG** command in Chapter 6 of this manual, and to the *PPCBug Firmware Package User's Manual*). If the check fails, a warning message displays. The calculated clock speed is also checked against known clock speeds and tolerances.

Memory Requirements

The debugger requires a total of 512KB of read/write memory. The debugger allocates this memory starting from the top of memory. For example, on a system which contains 64MB ($04000000) of read/write memory (i.e., DRAM), the debugger's memory page is located at $03F80000 to $03FFFFFF.

MPU, Hardware, and Firmware Initialization

The debugger performs the MPU, hardware, and firmware initialization process. This process occurs each time the PowerBase is reset or powered up. The steps below are a high-level outline; not all of the detailed steps are listed.

1. Sets MPU.MSR to known value.
2. Invalidates the MPU's data/instruction caches.
3. Clears all segment registers of the MPU.

4. Clears all block address translation registers of the MPU.
5. Initializes the MPU-bus-to-PCI-bus bridge device.
6. Initializes the PCI-bus-to-ISA-bus bridge device.
7. Calculates the external bus clock speed of the MPU.
8. Delays for 750 milliseconds.
9. Determines the CPU base board type.
10. Sizes the local read/write memory (i.e., DRAM).
11. Initializes the read/write memory controller. Sets base address of memory to $00000000.
12. Retrieves the speed of read/write memory from EEPROM.
13. Initializes the read/write memory controller with the speed of read/write memory.
14. Retrieves the speed of read only memory (i.e., Flash) from EEPROM.
15. Initializes the read only memory controller with the speed of read only memory.
16. Enables the MPU's instruction cache.
17. Copies the MPU's exception vector table from $FFF00000 to $00000000.
18. Verifies MPU type.
19. Verifies the external bus clock speed of the MPU.
20. Determines the debugger's console/host ports, and initializes the PC16550A.
21. Displays the debugger's copyright message.
22. Displays any hardware initialization errors that may have occurred.

23. Checksums the debugger object, and displays a warning message if the checksum failed to verify.
24. Displays the amount of local read/write memory found.
25. Verifies the configuration data that is resident in EEPROM, and displays a warning message if the verification failed.
26. Calculates and displays the MPU clock speed, verifies that the MPU clock speed matches the configuration data, and displays a warning message if the verification fails.
27. Displays the BUS clock speed, verifies that the BUS clock speed matches the configuration data, and displays a warning message if the verification fails.
28. Probes PCI bus for supported network devices.
29. Probes PCI bus for supported mass storage devices.
30. Initializes the memory/IO addresses for the supported PCI bus devices.
31. Executes Self-Test, if so configured. (Default is no Self-Test.)
32. Extinguishes the board fail LED, if Self-Test passed, and outputs any warning messages.
33. Executes boot program, if so configured. (Default is no boot.)
34. Executes the debugger monitor (i.e., issues the `PPC1-Bug>` prompt).

Multiprocessor Support (Remote Start)

The PowerPC board dual-port RAM feature makes the shared RAM available to remote processors as well as to the local processor. This can be done by either of the following two methods. Either method can be enabled/disabled by the **ENV** command as its Remote Start Switch method.

Note that PPCBug runs in single processor operation only.

Refer to Chapter 6 for information on setting the **ENV** command parameters.

Multiprocessor Control Register (MPCR) Method

A remote processor can initiate program execution in the local PowerPC board dual-port RAM by issuing a remote **GO** command using the Multiprocessor Control Register (MPCR). MPCR contains one of two words used to control communication between processors. The location of MPCR is calculated as local RAM size minus $1C000. The MPCR contents are organized as follows:

*	N/A	N/A	N/A

(MPCR)

The status codes stored in the MPCR are of two types:

- Status returned (from the monitor)
- Status set (by the bus master)

The status codes that may be returned from the monitor are:

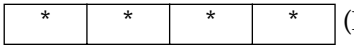
NUL ($00)	Wait; the initialization is not yet complete.
E ($45)	Code pointed to by the MPAR address is executing.
P ($50)	Program Flash Memory. The MPAR is set to the address of the Flash memory program control packet.
R ($52)	Ready; the firmware monitor is watching for a change.

You can only program Flash memory by the MPCR method. See the .PFLASH system call for a description of the Flash memory program control packet structure.

The status codes that may be set by the bus master are:

G ($47)	Initiate code at the MPAR address in a manner similar to the **GD** command.
B ($42)	Initiate code at the MPAR address, with breakpoints enabled, in a manner similar to the **GO** command.

The Multiprocessor Address Register (MPAR) contains the second of two words used to control communication between processors. The MPAR contents specify the address at which execution for the remote processor is to begin if the MPCR contains a **G** or **B**. The location of MPAR is calculated as MPCR plus 4. The MPAR is organized as follows:



(MPAR)

At power-up, the PPCBug self-test routines initialize RAM, including the memory locations used for multi-processor support (MPCR and MPAR).

The MPCR contains $00 at power-up, indicating that initialization is not yet complete. As the initialization proceeds, the execution path comes to the routine that displays the prompt. Before sending the prompt, this routine places an **R** in the MPCR to indicate that initialization is complete. Then the prompt is sent.

If no terminal is connected to the port, the MPCR is still polled to see whether an external processor requires control to be passed to the dual-port RAM. If a terminal does respond, the MPCR is polled for the same purpose while the serial port is being polled for user input.

A **G** placed in the MPCR by a remote processor indicates that the Go Direct type of transfer is requested (as with the **GD** command). A **B** in the MPCR indicates that breakpoints are to be armed before control is transferred (as with the **GO** command).

In either sequence, an **E** is placed in the MPCR to indicate that execution is underway just before control is passed to RAM. (Any remote processor could examine the MPCR contents.)

If the code being executed in dual-port RAM is to re-enter PPCBug, a system call using function $0063 (SYSCALL .RETURN) returns control to PPCBug with a new display prompt. Note that every time PPCBug returns to the prompt, an **R** is moved into the MPCR to indicate that control can be transferred once again to a specified RAM location.

Global Control and Status Register (GCSR) Methods

These methods support PPCBug for the PowerBase (MVME130*x*) when used as a WAN communications controller and known as PowerCom.

To configure a system with multiple PowerBase boards in a VME chassis, PowerBase memory must not by default automatically appear on the VMEbus at start-up, or overlap might occur; hence the need for a command to control this through the GCSR.

Initialization

1. At start-up, PPCBug initializes the PowerBase board hardware registers to configure the board environment according to a defined set of parameters. The specific parameters used may be default parameters or user selected parameters which were previously stored in non-volatile memory.
2. PPCBug initializes a residual data structure with certain important board-specific product data. The residual data structure adheres to the PRP Specification.
3. At start-up, PPCBug determines an assigned location in VMEbus short I/O space (A16) which the PowerBase GCSR registers occupy. The GCSR address is determined by the upper five software readable jumpers on header J2 (also referred to as the board select number).

   GCSR supports mapping from jumper settings. Short I/O addresses are contained in a table and looked up by comparing the board select number in the table to the jumpers present on the software readable jumper header J2. This function expects the first argument to be an unsigned integer representing the value read from the software readable jumper header J2. Refer to the section *Preparing the PowerBase* in Chapter 1.

Executing Remote Commands through the GCSR

Execution of commands issued from a remote processor through the VMEchip2 Global Control and Status Registers (GCSR) is performed as follows.

After performing the above initialization, PPCBug begins monitoring the GCSR and debugger input port. At this point, the host (or some other remote processor) may control execution of PPCBug via the GCSR or you may enter a debugger command on a terminal attached to the debug port. The host may invoke the commands listed below by writing to GCSR register(s) and then setting the SIG1 bit of the GCSR LM/SIG control register.

To execute a command through the GCSR on PowerBase:

1. Write the command code in GCSR0.
2. If the command requires, write information needed to GCSR1 through GCSR5.
3. Set SIG1 bit in LM/SIG register.
4. Allow sufficient time for the command to complete. Reading GCSR0 will show only the busy bit set (bit 07) once the command has been accepted, and will be cleared when the command has completed, unless the Selftest command was issued and has failed. Note that during the VME test portion of selftest, an attempt to read the GCSR will be unsuccessful.
5. Read GCSR to retrieve any information returned by the command.

Commands Available Through the GCSR Registers

For details, refer to the *PowerBase Embedded Controller Programmer's Reference Guide*.

Code	Command Name and Description
0x0001	Invoke Selftest (Execute Selftest)

This command sets a Busy/Failed Selftest bit and starts selftest. When the selftests are complete, it clears the bit that started the test. If the selftests pass, the Busy/Failed Selftest bit is cleared. If not, the address of an error message buffer is written to two GCSR registers, GCSR1 and GCSR2.

0x0002 Setup VMEbus Slave Decoder 1

This command sets Slave Decoder 1 starting, ending, translation, and select addresses, as well as write post and snoop control, using values found in the GCSR registers.

0x0004 Read (a PowerBase local resource)

This command reads data from an address found in two GCSR registers, and copies it into two other GCSR registers.

0x0008 Write (a PowerBase local resource)

This command copies data from two GCSR registers and writes it into the address found in two other GCSR registers.

0x0010 Request Board Information or query for residual data

This command returns the address of the board residual data. It will locate the start of free memory and initialize a residual data structure.

Using the GO Command through GCSR

A remote processor can initiate program execution in a local MVME130*x* PowerBase board's dual-port RAM by issuing a remote **GO** command using the Global Control and Status Registers (GCSR). The remote **GO** command causes the following sequence:

- Remote processor places the PowerBase execution address in general purpose registers 0 and 1 (GPCSR0 and GPCSR1)
- Remote processor sets bit 8 (SIG0) of the VMEchip2 LM/SIG register.
- PowerBase installs breakpoints and begins execution.

The result is identical to the MPCR method (with status code B) described in a previous section.

The GCSR registers are accessed in the VMEbus short I/O space. Each general purpose register is two bytes wide, occurring at an even address. The general purpose register number 0 is at an offset of $8 (local bus) or $4 (VMEbus) from the start of the GCSR registers. The local bus base address for the GCSR is variable. The VMEbus base address for the GCSR depends on the group select value and the module select value programmed in the Local Control and Status Registers (LCSR) of the PowerPC board.

The execution address is formed by reading the GCSR general purpose registers in the following manner:

GPCSR0	used as the upper 16 bits of the address
GPCSR1	used as the lower 16 bits of the address

The address appears as:

GPCSR0	GPCSR1

Data and Address Sizes

Data and address sizes are defined as follows:

A *byte* is eight bits, numbered 0 through 7, with bit 0 being the least significant.

A *half-word* is 16 bits, numbered 0 through 15, with bit 0 being the least significant.

A *word* is 32 bits, numbered 0 through 31, with bit 0 being the least significant.

Byte Ordering

The MPU on the PowerBase is programmed to big-endian byte ordering. Any attempt to use small-endian byte ordering immediately renders the PPCBug debugger unusable.

Performing Diagnostic Tests

The PPCBug hardware diagnostics are intended for testing and troubleshooting the PowerBase board.

In order to use the diagnostics, you must switch to the diagnostic directory. You may switch between directories by using the **SD** (Switch Directories) command. You may view a list of the commands in the directory that you are currently in by using the **HE** (Help) command.

If you are in the debugger directory, the debugger prompt `PPC1-Bug>` displays, and all of the debugger commands are available. Diagnostics commands cannot be entered at the `PPC1-Bug>` prompt.

If you are in the diagnostic directory, the diagnostic prompt `PPC1-Diag>` displays, and all of the debugger and diagnostic commands are available.

The diagnostic test groups are listed in the following table. Using the **HE** command, you can list the diagnostic routines available in each test group. Refer to the *PPC1Bug Diagnostics Manual* for complete descriptions of the diagnostic routines and instructions on how to invoke them.

Table 5-1. Diagnostic Test Groups

Test Group	Description
I82378	i82378 PCI/ISA Bridge Tests
PC16550	PC16550 UART Tests
RAM	Local RAM Tests
PCIBUS	PCI Bus/PMC Slot Tests

Notes 1. You may enter command names in either uppercase or lowercase.

2. Some diagnostics depend on restart defaults that are set up only in a particular restart mode. Refer to the documentation on a particular diagnostic for the correct mode.

5

Entering Debugging Commands

PPCBug is command-driven and performs its various operations in response to commands that you enter at the keyboard. When the debugger prompt `PPC1-Bug>` appears on the screen, then the debugger is ready to accept commands.

What you enter is stored in an internal buffer. Execution begins only after you press the Return key, allowing you to correct entry errors, if necessary, using the control characters described in the *PPCBug General Information* chapter.

After the debugger executes the command, the prompt reappears. However, if the command causes execution of user target code (for example **GO**) then control may or may not return to the debugger, depending on what the user program does. For example, if a breakpoint has been specified, then control returns to the debugger when the breakpoint is encountered during execution of the user program. Alternately, the user program could return to the debugger by means of the System Call Handler routine. .RETURN (described in the *PPCBug Firmware Package User's Manual*). For more about this, refer to the **GD**, **GO**, and **GT** command descriptions in the *PPCBug Firmware Package User's Manual*.

In general, a debugger command is made up of the following parts:

- The command name (e.g., **MD** or **md**). Note that either uppercase or lowercase is allowed.
- At least one intervening space before the first argument.
- Any required arguments, as specified by command.

- One or more options. Precede an option or a string of options with a semi-colon (**;**). If no option is entered, the command's default option conditions are used.

Conventions

The following conventions are used in the descriptions of the commands, arguments, and options that follow:

boldface strings	A boldface string is a literal such as a command name, program name, or option, and is to be typed just as it appears.
italic strings	An italic string is an argument.
|	A vertical bar separating two or more items indicates that a choice is to be made; only one of the items separated by this symbol should be selected.
[ ]	Square brackets enclose an item that is optional. The item may appear zero or one time.
{ }	Braces enclose an optional symbol that may occur zero or more times.

Command Arguments

The following arguments are common to many of the commands. Additional arguments are defined in the descriptions of commands found in the *PPCBug Firmware Package User's Manual*. You will also see the arguments for each command when you type **HE** to display the help menu.

EXP	Expression (see *EXP* on page 5-20).
ADDR	Address (see *ADDR* on page 5-22).
COUNT	Count; the syntax is the same as for *EXP* (see *EXP* on page 5-20).

RANGE	A range of memory addresses specified with a pair of arguments, either *ADDR ADDR* or *ADDR : COUNT*
TEXT	An ASCII string of up to 255 characters, delimited at each end by the single quote mark (').
PORT	Port Number (see *PORT* on page 5-23).

Use either a space or a comma to separate arguments. You may select the default value for an argument by inserting a pair of commas in place of the argument.

EXP

The EXP (expression) argument can be one or more numeric values separated by the arithmetic operators:

+	plus
-	minus
*	multiply by
/	divide by
&	logical AND
<<	shift left
>>	shift right

Numeric values may be expressed in either hexadecimal, decimal, octal, or binary by immediately preceding them with the proper base identifier.

Data Type	Base	Identifier	Examples
Integer	Hexadecimal	$	$FFFFFFFF
Integer	Decimal	&	&1974, &10-&4
Integer	Octal	@	@456
Integer	Binary	%	%1000110

If no base identifier is specified, then the numeric value is assumed to be hexadecimal.

A numeric value may also be expressed as a string literal of up to four characters. The string literal must begin and end with the single quote mark ('). The numeric value is interpreted as the concatenation of the ASCII values of the characters. This value is right-justified, as any other numeric value would be.

String Literal	Numeric Value (Hexadecimal)
'A'	41
'ABC'	414243
'TEST'	54455354

Evaluation of an expression is always from left to right unless parentheses are used to group part of the expression. There is no operator precedence. Subexpressions within parentheses are evaluated first. Nested parenthetical subexpressions are evaluated from the inside out.

Valid Expression Examples

Expression	Result (Hexadecimal)	Notes
FF0011	FF0011	
45+99	DE	
&45+&99	90	
@35+@67+@10	5C	
%10011110+%1001	A7	
88<<4	880	shift left
AA&F0	A0	logical AND

The total value of the expression must be between 0 and $FFFFFFFF.

ADDR

The syntax for the *ADDR* argument is similar to the syntax accepted by the PowerPC one-line assembler. All control addressing modes are allowed. An "address + offset register" mode is also provided.

ADDR Formats

The *ADDR* format is:

HexadecimalNumber {**[^S]** | **[^s]** | **[^U]** | **[^u]**} | **R***n*

Enter *ADDR* as a hexadecimal number (e.g., 20000 for address $00020000). The address, or starting address of a range, can be qualified by a suffix, either **^S** or **^s** for supervisor address space, or **^U** or **^u** for user address space. The default, when the suffix is not specified, is supervisor.

Once a qualifier has been entered, it remains valid for all addresses entered for that command sequence, until either the PPCBug is reentered or another qualifier is provided.

In the alternate register number (**R***n*) form, the debugger uses the address contained in MPU Register **R***n*, where *n* is 0 through 31 (i.e., 0, 1, . . . 31).

In commands with the address range specified as *ADDR ADDR*, and with size option **H** or **W** chosen, data at the second (ending) address is acted on only if the second address is a proper boundary for a half-word or word, respectively. Otherwise, the range is truncated so that the last byte acted upon is at an address that is a proper boundary.

Offset Registers

Eight pseudo-registers (Z0-Z7) called offset registers are used to simplify the debugging of relocatable and position-independent modules. The listing files in these types of programs usually start at an address (normally 0) that is not the one at which they are loaded, so it is harder to correlate addresses in the listing with addresses in the loaded program. The offset registers solve this problem by taking into account this difference and forcing the display of

addresses in a relative address+offset format. Offset registers have adjustable ranges and may even have overlapping ranges. The range for each offset register is set by two addresses: base and top. Specifying the base and top addresses for an offset register sets its range. In the event that an address falls in two or more offset registers' ranges, the one that yields the least offset is chosen.

Note Relative addresses are limited to 1MB (5 digits), regardless of the range of the closest offset register.

PORT

The *PORT* argument is the logical number of the port to be used to input or output. Valid port numbers which may be used for these commands are as follows:

0 or **00** Terminal port 0 (“console port”) is used for interactive user input and output (the default). This port number usually refers to the serial port labelled either COM1 or SER1 on the system board. However, this port number may also be used for the graphics adapter device.

1 or **01** Terminal port 10 (“host port”) is the default for downloading, uploading, concurrent mode, and transparent modes. This port is labeled either COM2 or SER2 on the system board.

Command Options

Many commands have one or more options, defined in the command descriptions found in the *PPCBug Firmware Package User’s Manual*. You will also see the options for each command when you type **HE** to display the help menu.

Precede an option or a string of options with a semi-colon (**;**). If no option is entered, the command’s default option conditions are used.

Terminal Input and Output Control Characters

You may use the following control codes for limited editing while entering commands at the PPC1-Bug> prompt:

CTRL-X (cancel line)	Move the cursor to the beginning of the line. If the terminal port is configured with the hardcopy or TTY option (refer to the **PF** command in the *PPCBug Firmware Package User's Manual*), then a **<CR><LF>** sequence is issued along with another prompt.
CTRL-H (backspace)	Moved the cursor back one position. The character at the new cursor position is erased. If the hardcopy option is selected, a "/" character is typed along with the deleted character.
DEL (delete or rubout) key	Performs the same function as **CTRL-H**.
CTRL-D (redisplay)	Redisplay the entire command line as entered so far is on the following line.
CTRL-A (repeat)	Repeat the previous line. This happens only at the command line. The last line entered is redisplayed but not executed. The cursor is positioned at the end of the line. You may enter the line as is or you can add more characters to it. You can edit the line by backspacing and typing over old characters.

The XON and XOFF characters in effect for the terminal port may be entered to control the output from any PPCBug command, if the XON/XOFF protocol is enabled (default). The characters listed are initialized by PPCBug, but you may change them with the **PF** command:

CTRL-S (wait)	Halt console output (XON).
CTRL-Q (resume)	Resume console output (XOFF).

PPCBug Debug Command Set

The PPCBug debugging commands are summarized in the following table. Arguments and options for the commands will be displayed when you use the **HE** (help) command. All command details are explained in the *PPCBug Firmware Package User's Manual*.

Table 5-2. Debugger Commands

Command Mnemonic	Command Title
AS	One Line Assembler
BC	Block of Memory Compare
BF	Block of Memory Fill
BI	Block of Memory Initialize
BM	Block of Memory Move
BR	Breakpoint Insert
NOBR	Breakpoint Delete
BS	Block of Memory Search
BV	Block of Memory Verify
CM	Concurrent Mode
NOCM	No Concurrent Mode
CNFG	Configure Board Information Block
CS	Checksum
DC	Data Conversion
DMA	Block of Memory Move
DS	One Line Disassembler
DU	Dump S-Records
ECHO	Echo String
ENV	Set Environment
GD	Go Direct (Ignore Breakpoints)
GEVBOOT	Global Environment Variable Boot (See Note)
GEVDEL	Global Environment Variable Delete (See Note)
GEVDUMP	Global Environment Variable(s) Dump (See Note)
GEVEDIT	Global Environment Variable Edit (See Note)
GEVINIT	Global Environment Variable Initialization (See Note)

Table 5-2. Debugger Commands (Continued)

Command Mnemonic	Command Title
GEVSHOW	Global Environment Variable(s) Display (See Note)
GN	Go to Next Instruction
GO	Go Execute User Program
GT	Go to Temporary Breakpoint
HE	Help
IOC	I/O Control for Disk (See Note)
IOI	I/O Inquiry (See Note)
IOP	I/O Physical (Direct Disk Access) (See Note)
IOT	I/O Teach for Configuring Disk Controller (See Note)
LO	Load S-Records from Host
MA	Macro Define/Display
NOMA	Macro Delete
MAE	Macro Edit
MAL	Enable Macro Listing
NOMAL	Disable Macro Listing
MAR	Load Macros
MAW	Save Macros
MD, MDS	Memory Display
MENU	System Menu
MM	Memory Modify
MMD	Memory Map Diagnostic
MS	Memory Set
MW	Memory Write
NAB	Automatic Network Boot (See Note)
NBH	Network Boot Operating System, Halt (See Note)
NBO	Network Boot Operating System (See Note)
NIOC	Network I/O Control (See Note)
NIOP	Network I/O Physical (See Note)
NIOT	Network I/O Teach (Configuration) (See Note)
NPING	Network Ping (See Note)

Table 5-2. Debugger Commands (Continued)

Command Mnemonic	Command Title
OF	Offset Registers Display/Modify
PA	Printer Attach
NOPA	Printer Detach
PBOOT	Bootstrap Operating System (See Note)
PF	Port Format
NOPF	Port Detach
PFLASH	Program Flash Memory
PS	Put RTC into Power Save Mode (See Note)
RB	ROMboot Enable
NORB	ROMboot Disable
RD	Register Display
REMOTE	Remote
RESET	Cold/Warm Reset
RL	Read Loop
RM	Register Modify
RS	Register Set
SD	Switch Directories
SET	Set Time and Date (See Note)
SYM	Symbol Table Attach
NOSYM	Symbol Table Detach
SYMS	Symbol Table Display/Search
T	Trace
TA	Terminal Attach
TIME	Display Time and Date (See Note)
TM	Transparent Mode
TT	Trace to Temporary Breakpoint
VE	Verify S-Records Against Memory
VER	Revision/Version Display
WL	Write Loop

Note Commands for Global Environment, I/O, networking, PBOOT, and RTC are not applicable to PowerBase.

Advanced Debugger Topics 6

Modifying Parameters in EEPROM

You can use the factory-installed debug monitor, PPCBug, to modify certain parameters contained in the PowerBase board's EEPROM.

- The Board Information Block in EEPROM contains various elements concerning operating parameters of the hardware. Use the PPCBug command **CNFG** to change those parameters.
- Use the PPCBug command **ENV** to change configurable PPCBug parameters in EEPROM.

The **CNFG** and **ENV** commands are both described in the *PPCBug Firmware Package User's Manual*. Refer to that manual for general information about their use and capabilities.

The following paragraphs present additional information about **CNFG** and **ENV** that is specific to the PPCBug debugger, along with the parameters that can be configured with the **ENV** command.

CNFG - Configure Board Information Block

Use this command to display and configure the Board Information Block, which is resident within the EEPROM. The Board Information Block contains various elements detailing specific operational parameters of the PowerBase board. The board structure for the PowerBase board is as shown in the following typical example:

```
Board (PWA) Serial Number    = “1234567                “
Board Identifier             = “POWERBASE                          ”
Artwork (PWA) Identifier     = “01-w3135F06A                       ”
MPU Clock Speed              = “067”
Bus Clock Speed              = “067”
Ethernet Address             = FFFFFFFFFFFF
Local SCSI Identifier        = “??”
System Serial Number         = “????????????????                  ”
System Identifier            = “????????????????????????????????????”
```


The parameters that are quoted are left-justified character (ASCII) strings padded with space characters, and the quotes (“) are displayed to indicate the size of the string. Parameters that are not quoted are considered data strings, and data strings are right-justified. The data strings are padded with zeroes if the length is not met.

The Board Information Block is factory-configured before shipment. There is no need to modify block parameters unless the EEPROM is corrupted.

Refer to the *PowerBase Embedded Controller Programmer’s Reference Guide* for the actual location and other information about the Board Information Block.

Refer to the *PPCBug Firmware Package User’s Manual* for a description of **CNFG** and examples.

ENV - Set Environment

Use the **ENV** command to view and/or configure interactively all PPCBug operational parameters that are kept in EEPROM.

Refer to the *PPCBug Firmware Package User’s Manual* for a description of the use of **ENV**. Additional information on control and status registers that affect these parameters is contained in the *PowerBase Embedded Controller Programmer’s Reference Guide*.

Configuring the PPCBug Parameters

Listed and described below are the parameters that you can configure using **ENV**. The default values shown were those in effect when this publication went to print. Note that where a prompt refers to NVRAM, the PowerBase uses EEPROM. Note also that some parameters are not applicable to PowerBase, e.g. those for SCSI and the automatic booting routines.

```
Bug or System environment [B/S] = B?
```

B Do not run the self test diagnostics during system start-up. Display the `PPC1-Bug>` prompt if the start-up fails or if you exit from the start-up sequence. (Default)

S Run the self test diagnostics during system start-up. Display the `PPC1-Diag>` prompt if the start-up fails or if you exit from the start-up sequence.

```
Field Service Menu Enable [Y/N] = N?
```

Y Display system menu in place of a debugger prompt if the start-up fails or if you exit from the start-up sequence.

N Display a debugger prompt if the start-up fails or if you exit from the start-up sequence. (Default)

```
Remote Start Method Switch [G/M/B/N] = B?
```

The method for executing a cross-loaded program when the PowerBase is cross-loaded from another VME-based CPU.

G Use the Global Control and Status Register (GCSR) method to pass and start execution of cross-loaded program (VMEchip2).

M Use the Multiprocessor Control Register (MPCR) method in shared RAM to pass and start execution of cross-loaded program.

B Use both the GCSR and the MPCR methods to pass and start execution of cross-loaded program.

N Do not use any remote start method.

```
Probe System for Supported I/O Controllers [Y/N] = Y?
```

Y Access the appropriate system buses (e.g., VMEbus, local MPU bus) to determine the presence of supported controllers. (Default)

N Do not access the VMEbus to determine the presence of supported controllers.

```
Auto-Initialize of NVRAM Header Enable [Y/N] =Y?
```

Y Enable auto-initialization of NVRAM header. (Default)

N Disable auto-initialization of NVRAM header.


```
Network PReP-Boot Mode Enable [Y/N] = N?
```

Y Enable network PReP-boot mode.

N Disable network PReP-boot mode. (Default)

```
Negate VMEbus SYSFAIL* Always [Y/N] = N?
```

Y Negate VMEbus SYSFAIL during board initialization.

N Negate VMEbus SYSFAIL after successful completion or entrance into PPCBug. (Default)

```
Local SCSI Bus Reset on Debugger Setup [Y/N] = N?
```

Y Reset the Local SCSI bus on debugger set-up.

N Do not reset the Local SCSI bus on debugger set-up. (Default)

```
Local SCSI Bus Negotiations Type [A/S/N] = A?
```

A Asynchronous SCSI bus negotiation

S Synchronous SCSI bus negotiation

N None.

```
Local SCSI Data Bus Width [W/N] = N?
```

W Wide SCSI (16-bit bus).

N Narrow SCSI (8-bit bus). (Default)

```
NVRAM Bootlist (GEV.fw-boot-path) Boot Enable [Y/N] = N?
```

Y Enable NVRAM Bootlist boot.

N Disable NVRAM Bootlist boot. (Default)

```
NVRAM Bootlist (GEV.fw-boot-path) Boot at power-up only [Y/N] = N?
```

Y Run NVRAM Bootlist boot at power-up reset only.

N Run NVRAM Bootlist boot at any reset. (Default)


```
NVRAM Bootlist (GEV.fw-boot-path) Boot Abort Delay = 5?
```

The time in seconds that the start-up sequence waits before starting NVRAM Bootlist boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value may be from 0-255.

```
Auto Boot Enable [Y/N]   = N?
```

Y Enable Auto Boot.

N Disable Auto Boot (Default).

```
Auto Boot at power-up only [Y/N] = N?
```

Y Run Auto Boot at power-up reset only.

N Run Auto Boot at any reset.

```
Auto Boot Scan Enable [Y/N] = Y?
```

Y Auto Boot boots from the devices in the Auto Boot scan Device Type List (Default)

N Auto Boot boots from the CLUN and DLUN

```
Auto Boot Scan Device Type List = FDISK/CDROM/TAPE/HDISK/?
```

The order in which Auto Boot selects available boot devices if Auto Boot Scan Enable is set to Y. The entries must be uppercase and be separated by back slashes (\).


```
Auto Boot Controller LUN   = 00?
```

The boot controller Logical Unit Number.

```
Auto Boot Device LUN      = 00?
```

The boot device Logical Unit Number.

```
Auto Boot Partition Number = 00?
```

The disk partition that the boot is run from, as specified in the *PowerPC Reference Platform* specification. The valid partitions are 1, 2, 3, or 4. If the parameter is set to 0, the firmware searches the partitions in order until it finds the first "bootable" partition.

```
Auto Boot Abort Delay    = 7?
```

The time in seconds that the start-up sequence waits before starting Auto Boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value may be from 0-255.

```
Auto Boot Default String [NULL for an empty string] = ?
```

A string (filename) which is passed on to the code being booted. The maximum length of this string is 16 characters.

```
ROM Boot Enable [Y/N] = N?
```

Y Enable ROM Boot

N Disable ROM Boot (Default)

```
ROM Boot at power-up only [Y/N] = Y?
```

Y Run ROM Boot at power-up only (Default)

N Run ROM Boot at any reset

```
ROM Boot Enable search of VMEbus [Y/N] = N?
```

Y Search the VMEbus address space for a ROM Boot module in addition to the normal areas of memory.

N VMEbus address space will not be accessed by ROM Boot.


```
ROM Boot Abort Delay    = 5?
```

The time in seconds that the start-up sequence waits before starting ROM Boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value may be from 0-255.

```
ROM Boot Direct Starting Address = FFF00000?
```

The first location tested when PPCBug searches for a ROM Boot module

```
ROM Boot Direct Ending Address   = FFFFFFFC?
```

The last location tested when PPCBug searches for a ROM Boot module

```
Network Auto Boot Enable [Y/N]   = N?
```

Y Enable Network Auto Boot

N Disable Network Auto Boot (Default)

```
Network Auto Boot at power-up only [Y/N] = Y?
```

Y Run Network Auto Boot at power-up reset only (Default)

N Run Network Auto Boot at any reset

```
Network Auto Boot Controller LUN = 00?
```

The boot controller Logical Unit Number.

```
Network Auto Boot Device LUN     = 00?
```

The boot device Logical Unit Number.

```
Network Auto Boot Abort Delay    = 5?
```

The time in seconds that the Network Auto Boot sequence waits before starting the boot. During the delay a user may exit to the debugger or diagnostics prompt by pressing the BREAK key. The value is from 0-255.

```
Network Auto Boot Configuration Parameters Offset (NVRAM)= 00001000?
```

The address where the network interface configuration parameters are saved/retained in EEPROM. These parameters are the necessary parameters to perform an unattended network boot. A typical offset might be $1000, but this is application-specific.

Note The default value of $00001000 locates the **NIOT** parameters just above the PRP partition of the EEPROM.

```
Memory Size Enable [Y/N]          = Y?
```

Y Memory will be sized for Self Test diagnostics. (Default)

N Memory will not be sized for Self Test diagnostics.

```
Memory Size Starting Address      = 00000000?
```

The Starting Address for memory sizing

```
Memory Size Ending Address        = 40000000?
```

The Ending Address for memory sizing. This is the calculated size of local memory. If the memory start is changed from $0, this parameter would also need to be adjusted. The default may be different for each board.


```
DRAM Speed in NANO Seconds = xx?
```

The default is set to the slowest speed found on the available banks in the DRAM memory used on the board., which varies depending on the speed of the DRAM memory.

```
ROM First Access Length (0 - 31) = x?
```

The number of clock cycles used in accessing the ROM. This is programmed into the MPC105 “ROMFAL” field (Memory Control Configuration Register 8: bits 23-27). Refer to Chapter 3 for appropriate values. The default ROMFAL value varies according to the bus clock speed for the system. The allowable range is from $0 to $1F.

```
ROM Next Access Length (0 - 15) = x?
```

The wait states in access time for nibble (or burst) mode accesses to bursting ROMs. This is programmed into the MPC105 “ROMNAL” field (Memory Control Configuration Register 8: bits 28-31). Refer to Chapter 3 for appropriate values. The default ROMNAL value varies according to the bus clock speed for the system. The allowable range is from $0 to $F.

```
DRAM Parity Enable [On-Detection/Always/Never] - O/A/N] = O?
```

O Enable DRAM parity on detection of a parity error.

A Enable DRAM parity always.

N Enable DRAM parity never.

```
L2Cache Parity Enable [On-Detection/Always/Never] - O/A/N] = O?
```

O Enable L2 Cache parity on detection of a parity error.

A Enable L2 Cache parity always.

N Enable L2 Cache parity never.


```
PCI Interrupts Route Control Registers (PIRQ0/1/2/3) = xxxxxxxx?
```

This parameter initializes the PIRQ*x* (PCI Interrupts) route control registers. The parameter is a 32-bit value divided into four parts, to yield the four PIRQ*x* (PIRQ0/1/2/3) route control registers, which are each a byte wide.

The default value, *xxxxxxxx*, is dependent upon the system type. Runs an **ENV;D** command to get the default value specific to the host system.

Configuring the VMEbus Interface

The following parameters set up the VMEbus interface for PowerBase modules. Refer to the VME2PCI and VMEchip2 ASICs information in Chapter 4 and to the *PowerBase Embedded Controller Programmer's Reference Guide* for information on these parameters. Figure 6-1 illustrates the processor-to-VMEbus mapping.

```
VME2PCI Master Master Enable [Y/N] = Y?
```

Y Enable all VMEbus interface master and slave decoders. (Default)

N Disable all VMEbus interface master and slave decoders.


```
VME2PCI Slave Enable #1 [Y/N] = Y?   = Y?
```

Y Enable the VME2PCI Slave Address Decoder #1.

N Disable VME2PCI Slave Address Decoder #1.

```
VME2PCI Slave Starting Address #1 = 01000000?
```

The starting address of the first PCI Memory Space for the VME2PCI's slave interface. PCI memory accesses within the range of the starting and ending addresses are passed on to the VMEchip2, as adjusted by the slave address offset. Only the upper 16 bits of this address are significant.

```
VME2PCI Slave Ending Address #1 = 1FFFFFFF?
```

The ending address of the first PCI Memory Space for the VME2PCI's slave interface. Only the upper 16 bits of this address are significant.

```
VME2PCI Slave Address Offset #1 = 00000000?
```

The offset used to translate the most significant 16 bits of the address to be presented to the VMEchip2 from the PCI bus. The address presented is equal to the sum of PCI address (bits 31-16) and the offset (bits 31-16). Bits 15-00 will be zero.

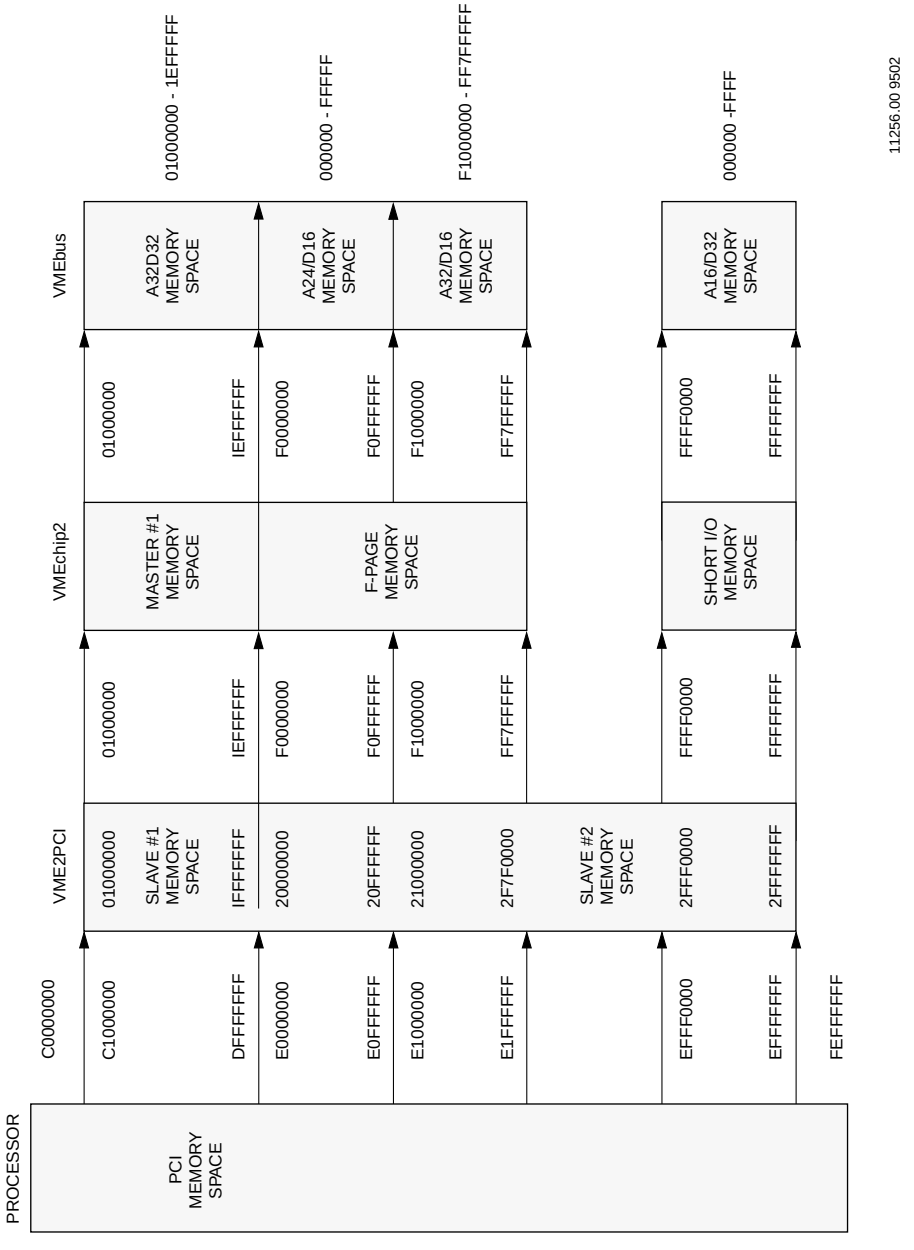


Figure 6-1. PPCBug Default Processor-to-VMEbus Mapping

```
VME2PCI Slave Enable #2 [Y/N] = Y?
```

Y Enable VME2PCI Slave Address Decoder #2.

N Disable VME2PCI Slave Address Decoder #2.

```
VME2PCI Slave Starting Address #2 = 20000000?
```

The starting address of the second PCI Memory Space for the VME2PCI's slave interface. PCI memory accesses within the range of the starting and ending addresses are passed on to the VMEchip2, as adjusted by the slave address offset. Only the upper 16 bits of this address are significant.


```
VME2PCI Slave Ending Address #2 = 2FFFFFFF?
```

The ending address of the second PCI Memory Space for the VME2PCI's slave interface. Only the upper 16 bits of this address are significant.

```
VME2PCI Slave Address Offset #2 = D0000000?
```

The offset used to translate the most significant 16 bits of the address to be presented to the VMEchip2 from the PCI bus. The address presented is equal to the sum of PCI address (bits 31-16) and the offset (bits 31-16). Bits 15-00 will be zero.

Configuring the Slave Address Decoders

The following parameters set up the two slave address decoders in the VMEchip2. The decoders are used to allow another VMEbus master to access a local resource of the PowerBase. Refer to the VMEchip2 ASICs information in Chapter 4 and the *PowerBase Embedded Controller Programmer's Reference Guide* for information on these parameters.

```
Slave Enable #1 [Y/N] = N?
```

Y Enable the Slave Address Decoder #1.

N Disable the Slave Address Decoder #1. (Default)

```
Slave Starting Address #1 = 00000000?
```

The base address of the local resource that is accessible by the VMEbus. (Default is the base of local memory)

```
Slave Ending Address #1   = 3FFFFFFF?
```

The ending address of the local resource that is accessible by the VMEbus. (Default is the end of calculated memory)

```
Slave Address Translation Address #1 = 80000000?
```

The base address of local resource that is associated with the slave starting and ending addresses. This allows the VMEbus address and the local address to be different.

```
Slave Address Translation Select #1 = 00000000?
```

A mask that defines which bits of the address are significant. A 1 indicates a significant bit. A 0 indicates a nonsignificant bit.

```
Slave Control #1 = 03FF?
```

The access restriction for the address space defined with this slave address decoder (VMEbus Slave Write Post and Snoop Control Register and VMEbus Slave Address Modifier Select Register). (Default = $03FF)

```
Slave Enable #2 [Y/N] = N?
```

Y Enable the Slave Address Decoder #2.

N Do not enable the Slave Address Decoder #2.

```
Slave Starting Address #2 = 00000000?
```

The base address of the local resource that is accessible by the VMEbus.

```
Slave Ending Address #2   = 00000000?
```

The ending address of the local resource that is accessible by the VMEbus.

```
Slave Address Translation Address #2 = 00000000?
```

The base address of local resource that is associated with the slave starting and ending addresses. This allows the VMEbus address and the local address to be different.

```
Slave Address Translation Select #2 = 00000000?
```

A mask that defines which bits of the address are significant. A 1 indicates a significant bit. A 0 indicates a nonsignificant bit.


```
Slave Control #2 = 0000?
```

The access restriction for the address space defined with this slave address decoder (VMEbus Slave Write Post and Snoop Control Register and VMEbus Slave Address Modifier Select Register).

```
Master Enable #1 [Y/N] = Y?
```

Y Enable the Master Address Decoder #1.

N Disable the Master Address Decoder #1.

```
Master Starting Address #1 = 00000000
```

The base address of the VMEbus resource that is accessible from the local bus. (Default is the end of calculated local memory)

```
Master Ending Address #1   = 1FFFFFFF?
```

The ending address of the VMEbus resource that is accessible from the local bus.

```
Master Control #1 = 0D?
```

The access characteristics for the address space defined with this master address decoder (Local Bus Slave Attribute Register). Default = $0D)

```
Master Enable #2 [Y/N] = N?
```

Y Enable the Master Address Decoder #2.

N Disable the Master Address Decoder #2. (Default)

```
Master Starting Address #2 = 00000000?
```

The base address of the VMEbus resource that is accessible from the local bus.

```
Master Ending Address #2 = 00000000?
```

The ending address of the VMEbus resource that is accessible from the local bus.


```
Master Control #2 = 00?
```

The access characteristics for the address space defined with this master address decoder (Local Bus Slave Attribute Register). (Default = $00)

```
Master Enable #3 [Y/N] = N?
```

Y Enable the Master Address Decoder #3.

N Disable the Master Address Decoder #3. (Default)

```
Master Starting Address #3 = 00000000?
```

The base address of the VMEbus resource that is accessible from the local bus.

```
Master Ending Address #3 = 00000000?
```

The ending address of the VMEbus resource that is accessible from the local bus.

```
Master Control #3 = 00?
```

The access characteristics for the address space defined with this master address decoder (Local Bus Slave Attribute Register).

```
Master Enable #4 [Y/N] = N?
```

Y Enable the Master Address Decoder #4.

N Disable the Master Address Decoder #4. (Default)

```
Master Starting Address #4 = 00000000?
```

The base address of the VMEbus resource that is accessible from the local bus.

```
Master Ending Address #4   = 00000000?
```

The ending address of the VMEbus resource that is accessible from the local bus.

```
Master Address Translation Address #4 = 00000000?
```

The base address of VMEbus resource that is associated with the starting and ending address selection from the previous questions. This allows the VMEbus address and the local address to be different.

```
Master Address Translation Select #4 = 00000000?
```

This register defines which bits of the address are significant. A logical 1 indicates significant address bits, logical 0 is nonsignificant.

```
Master Control #4 = 00?
```

The access characteristics for the address space defined with this master address decoder (Local Bus Slave Attribute Register).

```
Short I/O (VMEbus A16) Enable [Y/N] = Y?
```

Y Enable the Short I/O Address Decoder. (Default)

N Disable the Master Address Decoder.

```
Short I/O (VMEbus A16) Control      = 01?
```

The access characteristics for the address space defined with the Short I/O address decoder (Local Bus to VMEbus I/O Control Register)

```
F-Page (VMEbus A24) Enable [Y/N]    = Y?
```


Y Enable the F-Page Address Decoder. (Default)

N Disable the F-Page Address Decoder.

```
F-Page (VMEbus A24) Control         = 02?
```

The access characteristics for the address space defined with the F-Page address decoder (Local Bus to VMEbus I/O Control Register).

```
VMEC2 Vector Base #1          = 06?
```

The base interrupt vector (VMEchip2 Base Vector 0) for the component specified.

```
VMEC2 Vector Base #2          = 07?
```

The base interrupt vector (VMEchip2 Base Vector 1) for the component specified.

```
VMEC2 GCSR Group Base Address = D8?
```

The group address ($FFFF*xx*00) in Short I/O for this board.

```
VMEC2 GCSR Board Base Address = 00?.
```

The base address ($FFFFD8*x*0) in Short I/O for this board.

```
VMEbus Global Time Out Code   = 02?
```

The VMEbus time-out (VGTO) when the VMEchip2 is systems controller. (Default $02 = 256 µs)

```
VMEbus Access Time Out Code   = 02?
```

The local bus to VMEbus access time-out (VATO). (Default $02 = 32 ms)

Entering and Debugging Programs

There are various ways to enter a user program into system memory for execution. One way is to create the program using the **MM** (Memory Modify) command with the **DI** (assembler/disassembler) option, entering the program one source line at a time. After each source line is entered, it is assembled and the object code is loaded to memory. Refer to the *PPCBug Firmware Package User's Manual* for complete details of the PPCBug Assembler/Disassembler.

Another way is to download an object file from a host system. The program must be in S-record format (refer to the *PPCBug Firmware Package User's Manual*) and may have been assembled or compiled on the host system. Alternately, the program may have been previously created using the **MM** command as outlined above, and stored to the host using the **DU** (Dump) command. A communication link must exist between the host system and system board port 1. (Refer to Chapter 1 for hardware configuration information.) The file is downloaded from the host to system board memory by the **LO** (Load) command.

Once the object code has been loaded into memory, you can set breakpoints if desired and run the code or trace through it.

Calling System Routines from User Programs

Access to various PPCBug routines is provided via the System Call Handler. This gives a convenient way of doing character input/output and many other useful operations so that you do not have to write these routines into the target code. Refer to the *PPCBug Firmware Package User's Manual* for details on the routines available and how to invoke them from within a user program.

The System Call Handler is accessible through the **sc** (system call) instruction, with exception vector $00C00 (System Call Exception).

Preserving the Operating Environment

This section explains how to avoid contaminating the operating environment of the debugger. PPCBug uses certain portions of the system board onboard resources and also off-board system memory to contain temporary variables, exception vectors, etc. If you disturb resources upon which PPCBug depends, then the debugger may not function reliably or may not function at all.

If your application enables translation through the Memory Management Unit (MMU), and utilizes resources of the debugger (e.g., system calls), your application must create the necessary translation tables for the debugger to have access to its various resources. The debugger honors the enabling of the MMU; it does not alter or disable translation.

Memory Requirements

The debugger requires a total of 512K bytes of read/write memory. The debugger will allocate this memory from the top of memory. For example, on a system which contains 64 megabytes ($04000000) of read/write memory (i.e., DRAM), the debugger's memory page will be located at $03F80000 to $03FFFFFF.

This memory space is used by the debugger for program stack, I/O buffers, variables, and register files. If a user program is loaded (e.g., booted, S-Records) into memory, and if this program is utilizing the debugger's programmatic interface (i.e., system calls), the program must not modify this allocated memory.

Whenever the host hardware is reset, the following is done:

- Target IP (instruction pointer) is initialized to $00004000 (i.e., just above the memory space of the exception vector table).
- Target pseudo stack pointer is initialized to the starting location of the debugger's read/write memory space.
- Target IP will be set to the appropriate address if a program load operation.

Note that user programs should handle the stack area properly in that it should not write starting at the initialized location. Some compilers and assemblers may write to the stack prior to decrementing the stack.

This read/write memory space that is allocated for the debugger, by the debugger, may increase in future releases of the debugger. To properly compensate for the increased read/write memory requirements, user programs may utilize the target IP as indicator for the top (plus 1) of usable memory.

Exception Vectors Used by PPCBug

The following exception vectors are reserved for use by the debugger:

00100	System Reset	Used for the abort switch soft reset feature.
00700	Program	Used for instruction breakpoints.
00C00	System Call	Used for the System Call Handler.
02000	Run Mode	Used for instruction tracing.

These vectors may be taken over under a user's application. However, prior to returning control to the debugger these vectors must be restored for proper operation of the affected features.

MPU Registers

Certain MPU registers must be preserved for their specific uses.

MPU Register SPR275

MPU register SPR275 is reserved for use by the debugger. If SPR275 is to be used by the user program, it must be restored prior to utilizing debugger resources (system calls) and or returning control to the debugger.

MPU Registers SPR272-SPR274

These MPU registers are utilized by debugger as scratch registers.

Context Switching

Context switching is the switching from the debugger state to the user (target) state, or vice-versa. This switching occurs upon the invocation of either the **GD**, **GN**, **GO**, **GT**, **T**, or **TT** commands, or the return from user state to the debugger state.

User State to Debugger State

When the context switch transitions from the user state to the debugger state, the following MPUisters are captured:

R0-R31	General Purpose Registers
FR0-FR31	Floating Point Unit Data Registers
SR0-SR15	Segment Registers
SPR*n*	Special Purpose Registers (*n* is 1, 8, 9, 18, 19, 22, 25, 26, 27, 268, 269, 275, 282, 287, 528 - 543, 976 - 981, 1008, 1010)
IP	Instruction Pointer (copy of SPR26)
MSR	Machine State Register (copy of SPR27)
CR	Condition Register
FPSCR	Floating Point Status/Control Register

Debugger State to User State

When the context switch transitions from the debugger state to the user state, the following MPU registers are restored:

R0-R31	General Purpose Registers
FR0-FR31	Floating Point Unit Data Registers
SPR*n*	Special Purpose Registers (*n* is 1, 8, 9, 275, 1010)
IP	Instruction Pointer (copied to SPR26)
MSR	Machine State Register (copied to SPR2)
CR	Condition Register
FPSCR	Floating Point Status/Control Register

Note that on a restoration context switch, registers whose perspectives feature MMU characteristics and operating modes of the MPU are not restored. The debugger honors the user's MMU configuration. If the user's program wishes to utilize the programmatic interface (i.e., system calls) of the debugger, it must maintain the address translation of 1 to 1, and the I/O resources utilized by the debugger must be data cache inhibited.

Floating Point Support

The **MD** and **MM** commands allow display and modification of floating point data in memory. Use either the **MD** command or the **MM** command to assemble or disassemble floating point instructions.

Valid data types that can be used when modifying a floating point data register or a floating point memory location:

Integer Data Types	
Byte	12
Half-Word	1234
Word	12345678

Floating Point Data Types	
Single Precision Real	1_FF_7FFFFF
Double Precision Real	1_7FF_FFFFFFFFFFFF
Scientific Notation (decimal)	-3.12345678901234501_E+123

When entering data in single or double precision format, observe the following rules:

- The sign field is the first field and is a binary field.
- The exponent field is the second field and is a hexadecimal field.
- The mantissa field is the last field and is a hexadecimal field.
- The sign field, the exponent field, and at least the first digit of the mantissa field must be present (any unspecified digits in the mantissa field are set to zero).
- Each field must be separated from adjacent fields by an underscore.
- All the digit positions in the sign and exponent fields must be present.

Single Precision Real

The single precision real format would appear in memory as:

1-bit sign field	(1 binary digit)
8-bit biased exponent field	(2 hex digits. Bias = $7F)
23-bit fraction field	(6 hex digits)

A single precision number takes 4 bytes in memory.

Double Precision Real

The double precision real format would appear in memory as:

1-bit sign field	(1 binary digit)
11-bit biased exponent field	(3 hex digits. Bias = $3FF)
52-bit fraction field	(13 hex digits)

A double precision number takes 8 bytes in memory.

Note The single and double precision formats have an implied integer bit (always 1).

Scientific Notation

The scientific notation format provides a convenient way to enter and display a floating point decimal number. Internally, the number is assembled into a packed decimal number and then converted into a number of the specified data type.

Entering data in this format requires the following fields:

- An optional sign bit (+ or -)
- One decimal digit followed by a decimal point
- Up to 17 decimal digits (at least one must be entered)
- An optional Exponent field that consists of:
  - An optional underscore
  - The Exponent field identifier, letter E
  - An optional Exponent sign (+, -)
  - From 1 to 3 decimal digits

For more information about the floating point unit, refer to the *PowerPC 603 RISC Microprocessor User's Manual*.

Ordering Related Documentation

A

Motorola Computer Group Documents

The publications listed below are on related products, and some may be referenced in this document. If not shipped with this product, manuals may be purchased by contacting your local Motorola sales office.

Table A-1. Motorola Computer Group Documents

Document Title	Publication Number [1]
PowerBase Embedded Controller Installation and Use	VMEPBA/IH
PowerBase Embedded Controller Programmer's Reference Guide [2]	VMEPBA/PG
PowerCom Installation and Use Manual [2]	VMEPCOMA/IH
PPCBug Firmware Package User's Manual (Parts 1 and 2) [2]	PPCBUGA1/UM PPCBUGA2/UM
PPC1Bug Diagnostics Manual [2]	PPC1DIAA/UM
MVME762 Transition Module User's Manual [2]	VME762A/UM
SIM705 Serial Interface Module Installation Guide [2]	SIM705A/IH

Notes 1. Although not shown in the above list, each Motorola Computer Group manual publication number is suffixed with characters that represent the revision level of the document, such as "/*xx*2" (the second revision of a manual); a supplement bears the same number as the manual but has a suffix such as "/*xx*2A1" (the first supplement to the second revision of the manual).

2. Motorola documents marked with a [2] in the above list can be purchased as a set under part number **LK-PWRCOM**, when it becomes available. The content of this set may be revised as needed and without any notice to the customer.

Manufacturers' Documents

For additional information, refer to the following table for manufacturers' data sheets or user's manuals. As an additional help, a source for the listed document is also provided. Please note that in many cases, the information is preliminary and the revision levels of the documents are subject to change without notice.

To further assist your development effort, Motorola has collected some of the non-Motorola documents in this list from the suppliers. This bundle can be ordered as part number **68-PCIKIT**. The contents of this set is revised as needed and without any notice to the customer.

Table A-2. Manufacturers' Documents

Document Title and Source	Publication Number
PowerPC 603™ RISC Microprocessor Technical Summary Motorola Literature and Printing Distribution Services P.O. Box 20924 Phoenix, Arizona 85036-0924 Telephone: (602) 994-6561 FAX: (602) 994-6430	MPC603/D
PowerPC 603™ RISC Microprocessor User's Manual Motorola Literature and Printing Distribution Services P.O. Box 20924 Phoenix, Arizona 85036-0924 Telephone: (602) 994-6561 FAX: (602) 994-6430 OR IBM Microelectronics Mail Stop A25/862-1 PowerPC Marketing 1000 River Street Essex Junction, Vermont 05452-4299 Telephone: 1-800-PowerPC Telephone: 1-800-769-3772 FAX: 1-800-POWERfax FAX: 1-800-769-3732	MPC603UM/AD MPR603UMU-01

Table A-2. Manufacturers' Documents (Continued)

Document Title and Source	Publication Number
MPC105 PCI Bridge/Memory Controller User's Manual Motorola Literature and Printing Distribution Services P.O. Box 20924 Phoenix, Arizona 85036-0924 Telephone: (602) 994-6561 FAX: (602) 994-6430	MPC105UM/AD
PowerPC™ Microprocessor Family: The Programming Environments Motorola Literature and Printing Distribution Services P.O. Box 20924 Phoenix, Arizona 85036-0924 Telephone: (602) 994-6561 FAX: (602) 994-6430 OR IBM Microelectronics Mail Stop A25/862-1 PowerPC Marketing 1000 River Street Essex Junction, Vermont 05452-4299 Telephone: 1-800-PowerPC Telephone: 1-800-769-3772 FAX: 1-800-POWERfax FAX: 1-800-769-3732	MPCFPE/AD MPRPPCFPE-01
PC16550 UART National Semiconductor Corporation Customer Support Center (or nearest Sales Office) 2900 Semiconductor Drive P.O. Box 58090 Santa Clara, California 95052-8090 Telephone: 1-800-272-9959	PC16550DV
82378 System I/O (SIO) PCI-to-ISA Bridge Controller Intel Corporation Literature Sales P.O. Box 7641 Mt. Prospect, Illinois 60056-7641 Telephone: 1-800-548-4725	290473-003

Related Specifications

For additional information, refer to the following table for related specifications. As an additional help, a source for the listed document is also provided. Please note that in many cases, the information is preliminary and the revision levels of the documents are subject to change without notice.

Table A-3. Related Specifications

Document Title and Source	Publication Number
VME64 Specification VITA (VMEbus International Trade Association) 7825 E. Gelding Drive, Suite 104 Scottsdale, Arizona 85260-3415 Telephone: (602) 951-8866 FAX: (602) 951-0720 **NOTE:** An earlier version of this specification is available as:	ANSI/VITA 1-1994
Versatile Backplane Bus: VMEbus Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333 OR Microprocessor system bus for 1 to 4 byte data Bureau Central de la Commission Electrotechnique Internationale 3, rue de Varembé Geneva, Switzerland	ANSI/IEEE Standard 1014-1987 IEC 821 BUS
IEEE - Common Mezzanine Card Specification (CMC) Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333	P1386 Draft 2.0

Table A-3. Related Specifications (Continued)

Document Title and Source	Publication Number
IEEE - PCI Mezzanine Card Specification (PMC) Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333	P1386.1 Draft 2.0
Bidirectional Parallel Port Interface Specification Institute of Electrical and Electronics Engineers, Inc. Publication and Sales Department 345 East 47th Street New York, New York 10017-21633 Telephone: 1-800-678-4333	IEEE Standard 1284
Peripheral Component Interconnect (PCI) Local Bus Specification, Revision 2.0 PCI Special Interest Group P.O. Box 14070 Portland, Oregon 97214-4070 Marketing/Help Line Telephone: (503) 696-6111 Document/Specification Ordering Telephone: 1-800-433-5177or (503) 797-4207 FAX: (503) 234-6762	PCI Local Bus Specification
PowerPC Reference Platform (PRP) Specification, Third Edition, Version 1.0, Volumes I and II International Business Machines Corporation Power Personal Systems Architecture 11400 Burnet Rd. Austin, TX 78758-3493 Document/Specification Ordering Telephone: 1-800-PowerPC Telephone: 1-800-769-3772 Telephone: 708-296-9332	MPR-PPC-RPU-02

PowerBase Specifications B

Specifications

lists the general specifications for the PowerBase boards. The subsequent sections detail cooling requirements and FCC compliance.

A complete functional description of the PowerBase boards appears in Chapter 3. Specifications for the optional PMCs can be found in the documentation for those modules.

Table B-1. PowerBase Specifications

Characteristics		Specifications
MPU	MPC603, 66MHz	SPECint92 (estimated baseline/peak = 37.9/47.9)
		SPECint92 (estimated baseline/peak = 43.9/48.0
Memory	DRAM	8MB or 16MB with optional byte parity
	EEPROM	8KB
	PROM/Flash, or OTP ROM via optional mezzanine	1MB, or 4MB via optional mezzanine
Timers, via VMEchip2		One watchdog timer
		Two 32-bit tick timers
Power requirements, with no PMCs installed (See Note)		+5Vdc (±5%), 4.0A typical, 5.8A maximum +12Vdc , 0mA –12Vdc, 0mA (typical)
Operating temperature		0˚C to 55˚C entry air with forced-air cooling (refer to *Cooling Requirements* section)
Storage temperature		–40˚C to +85˚ C
Relative humidity		5% to 90% (non-condensing)
Vibration (operating)		2 Gs RMS, 20Hz-2000Hz random

Table B-1. PowerBase Specifications (Continued)

Characteristics		Specifications
Altitude (operating)		5000 meters (16,405 feet)
Physical dimensions (base board only)	Height	Double-high VME board, 9.2 in. (233 mm)
	Front panel width	0.8 in. (19.8 mm)
	Depth	6.3 in. (160 mm)
PCI Mezzanine Card (PMC) slots	Address/Data	A32/D32, PMC
	Bus Clock	Up to 33MHz
	Signaling	5V
	Power	7.5 watts maximum per slot (see Note)
	Module types	Basic, single-wide, front panel or P2 I/O (74.0 mm x 149.0 mm)
		Basic, double-wide, front panel or P2 I/O (149.0 mm x 149.0 mm)
	PMC I/O	Slot 1: Front panel and/or VMEbus P2 I/O
		Slot 2: Front panel I/O
Peripheral Computer Interface (PCI)	PCI bridge	MPC105
	PCIbus	32-bit, 33MHz
VMEbus ANSI/VITA 1-1994 VME64 (previously IEEE STD 1014)	DTB master	A32-D32; D08-D64, BLK
	DTB slave	A24-A32; D08-D64, BLK, UAT
	Arbiter	Round Robin or Priority
	Interrupt handler	IRQ 1-7
	Interrupt controller	Any one of seven
	System controller	Via jumper or auto detect
	Location monitor	Four LMA32

Note The power requirement listed for the MVME130*x* does not include the power requirements for the PMC slots. The PMC specification allows for 7.5 watts per PMC slot. The 15 watts total can be drawn from any combination of the four voltage sources provided by the PowerBase: +3.3V, +5V, +12V, and -12V.

Cooling Requirements

The Motorola PowerBase Embedded Controller is specified, designed, and tested to operate reliably with an incoming air temperature range from 0˚ to 55˚ C (32˚ to 131˚ F) with forced air cooling of the entire assembly (base board and modules) at a velocity typically achievable by using a 100 CFM axial fan. Temperature qualification is performed in a standard Motorola VMEsystem chassis. Twenty-five-watt load boards are inserted in two card slots, one on each side, adjacent to the board under test, to simulate a high power density system configuration. An assembly of three axial fans, rated at 100 CFM per fan, is placed directly under the VME card cage. The incoming air temperature is measured between the fan assembly and the card cage, where the incoming airstream first encounters the module under test. Test software is executed as the module is subjected to ambient temperature variations. Case temperatures of critical, high power density integrated circuits are monitored to ensure component vendors' specifications are not exceeded.

While the exact amount of airflow required for cooling depends on the ambient air temperature and the type, number, and location of boards and other heat sources, adequate cooling can usually be achieved with 10 CFM and 490 LFM flowing over the module. Less airflow is required to cool the module in environments having lower maximum ambients. Under more favorable thermal conditions, it may be possible to operate the module reliably at higher than 55˚ C with increased airflow. It is important to note that there are several factors, in addition to the rated CFM of the air mover, which determine the actual volume and speed of air flowing over a module.

FCC Compliance

The PowerBase was tested in an FCC-compliant chassis and meets the requirements for Class A equipment. FCC compliance was achieved under the following conditions:

- Shielded cables on all external I/O ports.
- Cable shields connected to earth ground via metal shell connectors bonded to a conductive module front panel.
- Conductive chassis rails connected to earth ground. This provides the path for connecting shields to earth ground.
- Front panel screws properly tightened.

For minimum RF emissions, it is essential that the conditions above be implemented. Failure to do so could compromise the FCC compliance of the equipment containing the module.

Regulatory Compliance

The PowerBase is a board level product and meant to be used in standard VME applications. As such, it is the responsibiility of the OEM to meet the regulatory guidelines as determined by its application.

All external I/O connectors are shielded to aid in meeting EMI emiassions standards. PowerBase is tested in an MCG chassis for EMI evaluation.

Connector Pin Assignments C

Introduction

This appendix summarizes the pin assignments for the following groups of interconnect signals for the PowerBase board:

Connector		Location	Table
VMEbus connector		P1	C-1
VMEbus connector, PMC Slot 1 for I/O		P2	C-2
Debug serial port, RJ45		J1	C-3
RISCwatch header		J3	C-4
CPU/debug connector		J4	C-5
PROM mezzanine connector		J8, J9	C-6
PMC connector	Slot 1 for 32-bit PCI	J11, J12	C-7
	Slot 2 for 32-bit PCI	J21, J22	C-8
	Slot 1 for P2 I/O	J14	C-9

Pin Assignments

The following tables furnish pin assignments only. For detailed descriptions of the various interconnect signals, consult the support information documentation for the MVME130*x* (contact your Motorola sales office) and/or the support information section of the MVME762 transition module and SIM705 documentation.

VMEbus Connector - P1

Two 96-pin connectors, P1 and P2, supply the interface between the base board and the VMEbus. P1 provides power and VME signals for 24-bit addressing and 16-bit data. Its pin assignments are set by the VMEbus specification. They are listed in the following table.

C

Table C-1. VMEbus Connector Pin Assignments, P1

	Row A	Row B	Row C	
1	VDO	VBBSY*	VD8	1
2	VD1	VBCLR*	VD9	2
3	VD2	VACFAIL*	VD10	3
4	VD3	VBGIN0*	VD11	4
5	VD4	VBGOUT0*	VD12	5
6	VD5	VBGIN1*	VD13	6
7	VD6	VBGOUT1*	VD14	7
8	VD7	VBGIN2*	VD15	8
9	GND	VBGOUT2*	GND	9
10	VSYSCLK	VBGIN3*	VSYSFAIL*	10
11	GND	VBGOUT3*	VBERR*	11
12	VDS1*	VBR0*	VSYSRESET*	12
13	VDS0*	VBR1*	VLWORD	13
14	VWRITE*	VBR2*	VAM5	14
15	GND	VBR3*	VA23	15
16	VDTACK*	VAM0	VA22	16
17	GND	VAM1	VA21	17
18	VAS*	VAM2	VA20	18
19	GND	VAM3	VA19	19
20	VIACK*	GND	VA18	20
21	VIACKIN*	VSERCLK	VA17	21
22	VIACKOUT*	VSERDAT	VA16	22
23	VAM4	GND	VA15	23
24	VA7	VIRQ7*	VA14	24
25	VA6	VIRQ6*	VA13	25
26	VA5	VIRQ5*	VA12	26
27	VA4	VIRQ4*	VA11	27
28	VA3	VIRQ3*	VA10	28
29	VA2	VIRQ2*	VA9	29
30	VA1	VIRQ1*	VA8	30
31	–12V	+5VSTDBY	+12V	31
32	+5V	+5V	+5V	32

VMEbus Connector - P2

Rows A and C of the P2 connector provide power and interface signals to the transition module, when one is used. Row B of P2 provides power to the PowerBase and the upper eight VMEbus lines as specified by the VMEbus specification.

Table C-2. VMEbus Connector Pin Assignments, P2

	Row A	Row B	Row C	
1	PMCIO2	+5V	PMCIO1	**1**
2	PMCIO4	GND	PMCIO3	**2**
3	PMCIO6	RETRY_	PMCIO5	**3**
4	PMCIO8	VA24	PMCIO7	**4**
5	PMCIO10	VA25	PMCIO9	**5**
6	PMCIO12	VA26	PMCIO11	**6**
7	PMCIO14	VA27	PMCIO13	**7**
8	PMCIO16	VA28	PMCIO15	**8**
9	PMCIO18	VA29	PMCIO17 PMCIO17F (See Note)	**9**
10	PMCIO20	VA30	PMCIO19	**10**
11	PMCIO22	VA31	PMCIO21	**11**
12	PMCIO24	GND	PMCIO23	**12**
13	PMCIO26	+5V	PMCIO25	**13**
14	PMCIO28	VD16	PMCIO27	**14**
15	PMCIO30	VD17	PMCIO29	**15**
16	PMCIO32	VD18	PMCIO31	**16**
17	PMCIO34	VD19	PMCIO33	**17**
18	PMCIO36	VD20	PMCIO35	**18**
19	PMCIO38	VD21	PMCIO37 PMCIO37F (See Note)	**19**
20	PMCIO40	VD22	PMCIO39	**20**
21	PMCIO42	VD23	PMCIO41	**21**
22	PMCIO44	VD24	PMCIO43	**22**
23	PMCIO46 PMCIO46F (See Note)	GND	PMCIO45	**23**

C

Table C-2. VMEbus Connector Pin Assignments, P2 (Continued)

24	PMCIO48	VD25	PMCIO47	**24**
25	PMCIO50	VD26	PMCIO49	**25**
26	PMCIO52	VD27	PMCIO51	**26**
27	PMCIO54	VD28	PMCIO53	**27**
28	PMCIO56	VD29	PMCIO55	**28**
			PMCIO55F (See Note)	
29	PMCIO58	VD30	PMCIO57	**29**
30	PMCIO60	VD31	PMCIO59	**30**
31	PMCIO62	GND	PMCIO61	**31**
32	PMCIO64	+5V	PMCIO63	**32**

Note Fused lines are used with the MVME762 transition board (factory configuration).

Debug Serial Port - J1

Table C-3. Debug Serial Port Pin Assignments, J1

	Signal
1	DCD_EXT*
2	RTS_EXT*
3	GND
4	TD_EXT
5	RD_EXT
6	GND
7	CTS_EXT*
8	DTR_EXT*

RISCwatch Header - J3

Table C-4. RISCwatch Header Pin Assignments, J3

Header Pin	MPC603 Pin (240-Pin QFP)	MPC603 I/O	Signal	Resistor
1	198	OUT	TDO	
2				
3	199	IN	TDI	4.7K pull-up
4	202	IN	TRST*	220 ohm pull-down
5				
6			3.3V	1K series
7	201	IN	TCK	4.7K pull-up
8				
9	200	IN	TMS	4.7K pull-up
10				
11	189	IN	SRESET*	4.7K pull-up
12				
13	213	IN	HRESET*	10K pull-up
14			KEY	
15	216	OUT	CHECKSTOP*	4.7K pull-up
16			GND	

Note The QACK_ signal on the MPC603 has a 1K-ohm pull-down resistor to allow the MPC603 processor to enter the state required for reading and writing SCAN string data.

CPU Connector - J4

One 190-pin SMT connector with center row of power and ground pins is used to provide access to the Processor Bus and some MPC105 signals. The pin assignments for this connector, J4, are as follows:

Table C-5. CPU Connector Pin Assignments, J4

1	PA0	GND	PA1	2
3	PA2		PA3	4
5	PA4		PA5	6
7	PA6		PA7	8
9	PA8		PA9	10
11	PA10		PA11	12
13	PA12		PA13	14
15	PA14		PA15	16
17	PA16		PA17	18
19	PA18		PA19	20
21	PA20		PA21	22
23	PA22		PA23	24
25	PA24		PA25	26
27	PA26		PA27	28
29	PA28		PA29	30
31	PA30		PA31	32
33	PAPAR0		PAPAR1	34
35	PAPAR2		PAPAR3	36
37	APE_		RSRV_	38

Table C-5. CPU Connector Pin Assignments, J4 (Continued)

39	PD0		PD1	**40**
41	PD2		PD3	**42**
43	PD4		PD5	**44**
45	PD6		PD7	**46**
47	PD8		PD9	**48**
49	PD10	**+5V**	PD11	**50**
51	PD12		PD13	**52**
53	PD14		PD15	**54**
55	PD16		PD17	**56**
57	PD18		PD19	**58**
59	PA20		PD21	**60**
61	PD22		PD23	**62**
63	PD24		PD25	**64**
65	PD26		PD27	**66**
67	PD28		PD29	**68**
69	PD30		PD31	**70**
71	PD32		PD33	**72**
73	PD34		PD35	**74**
75	PD36		PD37	**76**
77	PD38		PD39	**78**
79	PD40		PD41	**80**
81	PD42		PD43	**82**
83	PD44		PD45	**84**
85	PD46		PD47	**86**
87	PD48		PD49	**88**
89	PA50	**GND**	PD51	**90**
91	PD52		PD53	**92**
93	PD54		PD55	**94**
95	PD56		PD57	**96**
97	PD58		PD59	**98**
99	PD60		PD61	**100**
101	PD62		PD63	**102**
103	PDPAR0		PDPAR1	**104**
105	PDPAR2		PDPAR3	**106**
107	PDPAR4		PDPAR5	**108**
109	PDPAR6		PDPAR7	**110**
111				**112**
113	DPE_		DBDIS_	**114**

C

Table C-5. CPU Connector Pin Assignments, J4 (Continued)

115	TT0	**+3.3V**	TSIZ0	**116**
117	TT1		TSIZ1	**118**
119	TT2		TSIZ2	**120**
121	TT3		TC0	**122**
123	TT4		TC1	**124**
125	CI_		TC2	**126**
127	WT_		CSE0	**128**
129	GLOBAL_		CSE1	**130**
131	SHARED_		DBWO_	**132**
133	AACK_		TS_	**134**
135	ARTY_		XATS_	**136**
137	DRTY_		TBST_	**138**
139	TA_			**140**
141	TEA_			**142**
143			DBG_	**144**
145			DBB_	**146**
147			ABB_	**148**
149	TCLK_OUT		CPUGNT_	**150**
151	L2PRSNT_		CPUREQ_	**152**
153	L2ADSC_	**GND**	INT_	**154**
155	L2BAA_		MCPI_	**156**
157	L2DIRTYI_		SMI_	**158**
159	L2DIRTYO_		CKSTPI_	**160**
161	L2DOE_		CKSTPO_	**162**
163	L2WE_		HALTED	**164**
165	L2HIT_		TLBISYNC_	**166**
167	L2TALE		TBEN_	**168**
169	L2TALOE_		SUSPEND_	**170**
171	L2TOE_		DRVMOD0	**172**
173	L2TWE_		DRVMOD1	**174**
175	L2TV		NAPRUN	**176**
177	L2INT		QREQ_	**178**
179	SRESET_		QACK_	**180**
181	HRESET_		TDO	**182**
183	GND		TDI	**184**
185	CPUCLK1		TCK	**186**
187	CPUCLK2		TMS	**188**
189	CPUCLK3		TRST_	**190**

PROM Mezzanine Connectors - J8 and J9

C

Two 64-pin surface-mount connectors, J8 and J9, supply the interface between the add-on PROM mezzanine and the PowerBase. The pin assignments are listed in the following two tables.

Table C-6. PROM Mezzanine Connectors Pin Assignments, J8 and J9

	J8				J9		
1	GND	FA0	2	1	BMD21	GND	**2**
3	FA1	FA2	4	3	GND	BMD22	**4**
5	FA3	GND	6	5	BMD23	BMD24	**6**
7	GND	FA4	8	7	BMD25	GND	**8**
9	FA5	FA6	10	9	GND	BMD26	**10**
11	FA7	GND	12	11	BMD27	BMD28	**12**
13	GND	FA8	14	13	BMD29	+5V	**14**
15	FA9	FA10	16	15	+5V	BMD30	**16**
17	FA11	GND	18	17	BMD31	BMD32	**18**
19	GND	FA12	20	19	BMD33	GND	**20**
21	FA13	FA14	22	21	GND	BMD34	**22**
23	FA15	GND	24	23	BMD35	BMD36	**24**
25	GND	FA16	26	25	BMD37	+5V	**26**
27	FA17	FA18	28	27	+5V	BMD38	**28**
29	FA19	GND	30	29	BMD39	BMD40	**30**
31	GND	FOE_RCS1_L	32	31	BMD41	GND	**32**
33	FNR	BMD0	34	33	GND	BMD42	**34**
35	BMD1	GND	36	35	BMD43	BMD44	**36**
37	GND	BMD2	38	37	BMD45	+5V	**38**
39	BMD3	BMD4	40	39	+5V	BMD46	**40**
41	BMD5	GND	42	41	BMD47	BMD48	**42**
43	GND	BMD6	44	43	BMD49	GND	**44**
45	BMD7	BMD8	46	45	GND	BMD50	**46**
47	BMD9	GND	48	47	BMD51	BMD52	**48**
49	GND	BMD10	50	49	BMD53	GND	**50**
51	BMD11	BMD12	52	51	GND	BMD54	**52**
53	BMD13	GND	54	53	BMD55	BMD56	**54**
55	GND	BMD14	56	55	BMD57	GND	**56**
57	BMD15	BMD16	58	57	GND	BMD58	**58**
59	BMD17	GND	60	59	BMD59	BMD60	**60**
61	GND	BMD18	62	61	BMD61	GND	**62**
63	BMD19	BMD20	64	63	BMD62	BMD63	**64**

PCI Mezzanine Card Connectors - J11 and J12

Two 64-pin connectors, J11 and J12, supply the interface between the PowerBase board and an optional PCI mezzanine card (PMC). The pin assignments for PMC Slot 1 are listed in the following table.

Table C-7. PMC1 Connector Pin Assignments, J11 and J12

	J11				J12		
1	TCK	-12V	2	1	+12V	TRST_	**2**
3	GND	INTA_	4	3	TMS	TDO	**4**
5	INTB_	INTC_	6	5	TDI	GND	**6**
7	PMCPRSNT1_	+5V	8	7	GND	Not Used	**8**
9	INTD_	Not Used	10	9	Not Used	Not Used	**10**
11	GND	Not Used	12	11	Pull-up	+3.3V	**12**
13	CLK	GND	14	13	RST_	Pull-down	**14**
15	GND	PMCGNT1_	16	15	+3.3V	Pull-down	**16**
17	PMCREQ1_	+5V	18	17	Not Used	GND	**18**
19	+5V	AD31	20	19	AD30	AD29	**20**
21	AD28	AD27	22	21	GND	AD26	**22**
23	AD25	GND	24	23	AD24	+3.3V	**24**
25	GND	C/BE3_	26	25	IDSEL1	AD23	**26**
27	AD22	AD21	28	27	+3.3V	AD20	**28**
29	AD19	+5V	30	29	AD18	GND	**30**
31	+5V	AD17	32	31	AD16	C/BE2_	**32**
33	FRAME_	GND	34	33	GND	Not Used	**34**
35	GND	IRDY_	36	35	TDRY_	+3.3V	**36**
37	DEVSEL_	+5V	38	37	GND	STOP_	**38**
39	GND	LOCK_	40	39	PERR_	GND	**40**
41	SDONE_	SBO_	42	41	+3.3V	SERR_	**42**
43	PAR	GND	44	43	C/BE1_	GND	**44**
45	+5V	AD15	46	45	AD14	AD13	**46**

C

Table C-7. PMC1 Connector Pin Assignments, J11 and J12 (Continued)

47	AD12	AD11	48	47	GND	AD10	**48**
49	AD09	+5V	50	49	AD08	+3.3V	**50**
51	GND	C/BE0_	52	51	AD07	Not Used	**52**
53	AD06	AD05	54	53	+3.3V	Not Used	**54**
55	AD04	GND	56	55	Not Used	GND	**56**
57	+5V	AD03	58	57	Not Used	Not Used	**58**
59	AD02	AD01	60	59	GND	Not Used	**60**
61	AD00	+5V	62	61	ACK64_	+3.3V	**62**
63	GND	REQ64_	64	63	GND	Not Used	**64**

PCI Mezzanine Card Connectors - J21 and J22

Two 64-pin connectors, J21 and J22, supply the interface between the PowerBase board and an optional PCI mezzanine card (PMC). The pin assignments for PMC Slot 2 are listed in the following table.

Table C-8. PMC2 Connector Pin Assignments, J21 and J22

J21				J22			
1	TCK	-12V	2	1	+12V	TRST_	**2**
3	GND	INTA_	4	3	TMS	TDO	**4**
5	INTB_	INTC_	6	5	TDI	GND	**6**
7	PMCPRSNT2_	+5V	8	7	GND	Not Used	**8**
9	INTD_	Not Used	10	9	Not Used	Not Used	**10**
11	GND	Not Used	12	11	Pull-up	+3.3V	**12**
13	CLK	GND	14	13	RST_	Pull-down	**14**
15	GND	PMCGNT2_	16	15	+3.3V	Pull-down	**16**
17	PMCREQ2_	+5V	18	17	Not Used	GND	**18**
19	+5V	AD31	20	19	AD30	AD29	**20**
21	AD28	AD27	22	21	GND	AD26	**22**
23	AD25	GND	24	23	AD24	+3.3V	**24**
25	GND	C/BE3_	26	25	IDSEL2	AD23	**26**
27	AD22	AD21	28	27	+3.3V	AD20	**28**
29	AD19	+5V	30	29	AD18	GND	**30**
31	+5V	AD17	32	31	AD16	C/BE2_	**32**
33	FRAME_	GND	34	33	GND	Not Used	**34**
35	GND	IRDY_	36	35	TDRY_	+3.3V	**36**
37	DEVSEL_	+5V	38	37	GND	STOP_	**38**
39	GND	LOCK_	40	39	PERR_	GND	**40**
41	SDONE_	SBO_	42	41	+3.3V	SERR_	**42**
43	PAR	GND	44	43	C/BE1_	GND	**44**
45	+5V	AD15	46	45	AD14	AD13	**46**
47	AD12	AD11	48	47	GND	AD10	**48**

C

Table C-8. PMC2 Connector Pin Assignments, J21 and J22 (Continued)

49	AD09	+5V	50	49	AD08	+3.3V	**50**
51	GND	C/BE0_	52	51	AD07	Not Used	**52**
53	AD06	AD05	54	53	+3.3V	Not Used	**54**
55	AD04	GND	56	55	Not Used	GND	**56**
57	+5V	AD03	58	57	Not Used	Not Used	**58**
59	AD02	AD01	60	59	GND	Not Used	**60**
61	AD00	+5V	62	61	ACK64_	+3.3V	**62**
63	GND	REQ64_	64	63	GND	Not Used	**64**

PCI Mezzanine Card I/O Connector - J14

The mapping of the PMC Slot 1 I/O connector J14 to the VMEbus P2 connector is as follows:

Table C-9. PMC1 Connector Pin Assignments for I/O, J14 and VMEbus P2

J14	VMEbus P2
1	C1
3	C2
5	C3
7	C4
9	C5
11	C6
13	C7
15	C8
17	C9
19	C10
21	C11
23	C12
25	C13
27	C14
29	C15
31	C16
33	C17
35	C18
37	C19
39	C20
41	C21
43	C22
45	C23
47	C24
49	C25
51	C26
53	C27
55	C28
57	C29
59	C30
61	C31
63	C32

J14	VMEbus P2
2	A1
4	A2
6	A3
8	A4
10	A5
12	A6
14	A7
16	A8
18	A9
20	A10
22	A11
24	A12
26	A13
28	A14
30	A15
32	A16
34	A17
36	A18
38	A19
40	A20
42	A21
44	A22
46	A23
48	A24
50	A25
52	A26
54	A27
56	A28
58	A29
60	A30
62	A31
64	A32

Glossary

Abbreviations, Acronyms, and Terms to Know

This glossary defines some of the abbreviations, acronyms, and key terms used in this document.

10Base5 See thick Ethernet.

10Base2 See thin Ethernet.

10BaseT See twisted-pair Ethernet.

ACIA **A**synchronous **C**ommunications **I**nterface **A**dapter

AIX **A**dvanced **I**nteractive e**X**ecutive (IBM version of UNIX)

architecture The main overall design in which each individual hardware component of the computer system is interrelated. The most common uses of this term are 8-bit, 16-bit, or 32-bit architectural design systems.

ASCII **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange, a 7-bit code used to encode alphanumeric information. In the IBM-compatible world, this is expanded to 8 bits to encode a total of 256 alphanumeric and control characters.

ASIC **A**pplication-**S**pecific **I**ntegrated **C**ircuit

AUI **A**ttachment **U**nit **I**nterface

BBRAM Battery Backed-up Random Access Memory

bi-endian Having big-endian and little-endian byte ordering capability.

big-endian A byte-ordering method in memory where the address *n* of a word corresponds to the most significant byte. In an addressed memory word, the bytes are ordered (left to right) 0, 1, 2, 3, with 0 being the most significant byte.

BIOS **B**asic **I**nput/**O**utput **S**ystem. The built-in program that controls the basic functions of communications between the processor and the I/O devices (peripherals). Also referred to as ROM BIOS.

BitBLT **Bit** Boundary **BL**ock **T**ransfer. A type of graphics drawing routine that moves a rectangle of data from one area of display memory to another. The data need not have any particular alignment.

BLT **BL**ock **T**ransfer

board The term more commonly used to refer to a PCB (printed circuit board). Basically, a flat board made of nonconducting material, such as plastic or fiberglass, on which chips and other electronic components are mounted. Also referred to as a circuit board or card.

bpi **b**its **p**er **i**nch

bps **b**its **p**er **s**econd

bus The pathway used to communicate between the CPU, memory, and various input/output devices, including floppy drives and hard disk drives. Available in various widths (8-, 16-, and 32-bit), with accompanying increases in speed.

cache A high-speed memory that resides logically between a central processing unit (CPU) and the main memory. This temporary memory holds the data and/or instructions that the CPU is most likely to use over and over again and avoids frequent accesses to the slower hard drive or floppy disk drive.

CAS **C**olumn **A**ddress **S**trobe. The clock signal used in dynamic RAMs to control the input of column addresses.

CD **C**ompact **D**isc. A hard, round, flat portable storage unit that stores information digitally.

CD-ROM **C**ompact **D**isk **R**ead-**O**nly **M**emory

CFM **C**ubic **F**eet per **M**inute

CISC **C**omplex-**I**nstruction-**S**et **C**omputer. A computer whose processor is designed to sequentially run variable-length instructions, many of which require several clock cycles, that perform complex tasks and thereby simplify programming.

CODEC **CO**der/**DEC**oder

Color Difference (CD) The signals of (R-Y) and (B-Y) without the luminance (-Y) signal. The Green signals (G-Y) can be extracted by these two signals.

Composite Video Signal (CVS/CVBS) Signal that carries video picture information for color, brightness and synchronizing signals for both horizontal and vertical scans. Sometimes referred to as "Baseband Video".

cpi **c**haracters **p**er **i**nch

cpl **c**haracters **p**er **l**ine

CPU **C**entral **P**rocessing **U**nit. The master computer unit in a system.

DCE **D**ata **C**ircuit-terminating **E**quipment.

DLL **D**ynamic **L**ink **L**ibrary. A set of functions that are linked to the referencing program at the time it is loaded into memory.

DMA **D**irect **M**emory **A**ccess. A method by which a device may read or write to memory directly without processor intervention. DMA is typically used by block I/O devices.

DOS **D**isk **O**perating **S**ystem

dpi **d**ots **p**er **i**nch

DRAM **D**ynamic **R**andom **A**ccess **M**emory. A memory technology that is characterized by extremely high density, low power, and low cost. It must be more or less continuously refreshed to avoid loss of data.

DTE **D**ata **T**erminal **E**quipment.

ECC **E**rror **C**orrection **C**ode

ECP **E**xtended **C**apability **P**ort

EEPROM — **E**lectrically **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory. A memory storage device that can be written repeatedly with no special erasure fixture. EEPROMs do not lose their contents when they are powered down.

EISA (bus) — **E**xtended **I**ndustry **S**tandard **A**rchitecture (bus) (IBM). An architectural system using a 32-bit bus that allows data to be transferred between peripherals in 32-bit chunks instead of the 16-bit or 8-bit units that most systems use. With the transfer of larger bits of information, the machine is able to perform much faster than the standard ISA bus system.

EPP — **E**nhanced **P**arallel **P**ort

EPROM — **E**rasable **P**rogrammable **R**ead-**O**nly **M**emory. A memory storage device that can be written once (per erasure cycle) and read many times.

ESCC — **E**nhanced **S**erial **C**ommunication **C**ontroller

ESD — **E**lectro-**S**tatic **D**ischarge/Damage

Ethernet — A local area network standard that uses radio frequency signals carried by coaxial cables.

FDC — **F**loppy **D**isk **C**ontroller

FDDI — **F**iber **D**istributed **D**ata **I**nterface. A network based on the use of optical-fiber cable to transmit data in non-return-to-zero, invert-on-1s (NRZI) format at speeds up to 100 Mbps.

FIFO — **F**irst-**I**n, **F**irst-**O**ut. A memory that can temporarily hold data so that the sending device can send data faster than the receiving device can accept it. The sending and receiving devices typically operate asynchronously.

firmware — The program or specific software instructions that have been more or less permanently burned into an electronic component, such as a ROM (read-only memory) or an EPROM (erasable programmable read-only memory).

frame — One complete television picture frame consists of 525 horizontal lines with the NTSC system. One frame consists of two Fields.

graphics controller On EGA and VGA, a section of circuitry that can provide hardware assistance for graphics-drawing algorithms by performing logical functions on data written to display memory.

HAL **H**ardware **A**bstraction **L**ayer. The lower-level hardware interface module of the Windows NT operating system. It contains platform-specific functionality.

hardware The term used to describe any of the physical embodiments of a computer system, with emphasis on the electronic circuits (the computer) and electromechanical devices (peripherals) that make up the system. A computing system is normally spoken of as having two major components: hardware and software.

HCT **H**ardware **C**onformance **T**est. A test used to ensure that both hardware and software conform to the Windows NT interface.

I/O **I**nput/**O**utput

IBC PCI/**I**SA **B**ridge **C**ontroller

IDE **I**ntelligent **D**evice **E**xpansion

IEEE **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers

interlaced A graphics system in which the even scanlines are refreshed in one vertical cycle (field), and the odd scanlines are refreshed in another vertical cycle. Its advantage is that the video bandwidth is roughly half that required for a non-interlaced system of the same resolution. This results in less costly hardware and may also make it possible to display a resolution that would otherwise be impossible on given hardware. The disadvantage of an interlaced system is flicker, especially when displaying objects that are only a few scanlines high.

IQ Signals Similar to the color difference signals (R-Y), (B-Y) but using different vector axis for encoding or decoding. Used by some USA TV and IC manufacturers for color decoding.

ISA (bus)	**I**ndustry **S**tandard **A**rchitecture (bus). The de facto standard system bus for IBM-compatible computers until the introduction of VESA and PCI. Used in the reference platform specification. (IBM)
ISASIO	**ISA S**uper **I**nput/**O**utput device
ISDN	**I**ntegrated **S**ervices **D**igital **N**etwork. A standard for digitally transmitting video, audio, and electronic data over public phone networks.
LAN	**L**ocal **A**rea **N**etwork
LED	**L**ight-**E**mitting **D**iode
LFM	**L**inear **F**eet per **M**inute
little-endian	A byte-ordering method in memory where the address *n* of a word corresponds to the least significant byte. In an addressed memory word, the bytes are ordered (left to right) 3, 2, 1, 0, with 3 being the most significant byte.
MBLT	**M**ultiplexed **BL**ock **T**ransfer
MCA (bus)	**M**icro **C**hannel **A**rchitecture
MCG	**M**otorola **C**omputer **G**roup
MFM	**M**odified **F**requency **M**odulation
MIDI	**M**usical **I**nstrument **D**igital **I**nterface. The standard format for recording, storing, and playing digital music.
MPC	**M**ultimedia **P**ersonal **C**omputer
MPC105	The PowerPC-to-PCI bus bridge chip developed by Motorola for the Ultra 603/Ultra 604 system board. It provides the necessary interface between the MPC603/MPC604 processor and the Boot ROM (secondary cache), the DRAM (system memory array), and the PCI bus.
MPC601	Motorola's component designation for the PowerPC 601 microprocessor.
MPC603	Motorola's component designation for the PowerPC 603 microprocessor.
MPC603e	Motorola's component designation for the PowerPC 603e microprocessor.

MPC604	Motorola's component designation for the PowerPC 604 microprocessor.
MPU	**M**icro**P**rocessing **U**nit
MTBF	**M**ean **T**ime **B**etween **F**ailures. A statistical term relating to reliability as expressed in power-on hours (poh). It was originally developed for the military and can be calculated several different ways, yielding substantially different results. The specification is based on a large number of samplings in one place, running continuously, and the rate at which failure occurs. MTBF is not representative of how long a device or any individual component is likely to last, nor is it a warranty, but rather an indicator of the relative reliability of a family of products.
multisession	The ability to record additional information, such as digitized photographs, on a CD-ROM after a prior recording session has ended.
non-interlaced	A video system in which every pixel is refreshed during every vertical scan. A non-interlaced system is normally more expensive than an interlaced system of the same resolution, and is usually said to have a more pleasing appearance.
nonvolatile memory	A memory in which the data content is maintained whether the power supply is connected or not.
NTSC	**N**ational **T**elevision **S**tandards **C**ommittee (USA)
NVRAM	**N**on-**V**olatile **R**andom **A**ccess **M**emory
OEM	**O**riginal **E**quipment **M**anufacturer
OMPAC	**O**ver-**M**olded **P**ad **A**rray **C**arrier
OS	**O**perating **S**ystem. The software that manages the computer resources, accesses files, and dispatches programs.
OTP	**O**ne-**T**ime **P**rogrammable
palette	The range of colors available on the screen, not necessarily simultaneously. For VGA, this is either 16 or 256 simultaneous colors out of 262,144.

parallel port A connector that can exchange data with an I/O device eight bits at a time. This port is more commonly used for the connection of a printer to a system.

PCI (local bus) **P**eripheral **C**omponent **I**nterconnect (local bus) (Intel). A high-performance, 32-bit internal interconnect bus used for data transfer to peripheral controller components, such as those for audio, video, and graphics.

PCMCIA (bus) **P**ersonal **C**omputer **M**emory **C**ard **I**nternational **A**ssociation (bus). A standard external interconnect bus which allows peripherals adhering to the standard to be plugged in and used without further system modification.

PDS **P**rocessor **D**irect **S**lot

physical address A binary address that refers to the actual location of information stored in secondary storage.

PIB **P**CI-to-**I**SA **B**ridge

pixel An acronym for picture element, also called a pel. A pixel is the smallest addressable graphic on a display screen. In RGB systems, the color of a pixel is defined by some Red intensity, some Green intensity, and some Blue intensity.

PLL **P**hase-**L**ocked **L**oop

PMC **P**CI **M**ezzanine **C**ard

POWER **P**erformance **O**ptimized **W**ith **E**nhanced **R**ISC architecture (IBM)

PowerPC™ The trademark used to describe the **P**erformance **O**ptimized **W**ith **E**nhanced **R**ISC microprocessor architecture for **P**ersonal **C**omputers developed by the IBM Corporation. PowerPC is superscalar, which means it can handle more than one instruction per clock cycle. Instructions can be sent simultaneously to three types of independent execution units (branch units, fixed-point units, and floating-point units), where they can execute concurrently, but finish out of order. PowerPC is used by Motorola, Inc. under license from IBM.

PowerPC 601™ The first implementation of the PowerPC family of microprocessors. This CPU incorporates a memory management unit with a 256-entry buffer and a 32KB unified (instruction and data) cache. It provides a 64-bit data bus and a separate 32-bit address bus. PowerPC 601 is used by Motorola, Inc. under license from IBM.

PowerPC 603™ The second implementation of the PowerPC family of microprocessors. This CPU incorporates a memory management unit with a 64-entry buffer and an 8KB (instruction and data) cache. It provides a selectable 32-bit or 64-bit data bus and a separate 32-bit address bus. PowerPC 603 is used by Motorola, Inc. under license from IBM.

PowerPC 603e™ A variant of the second implementation of the PowerPC family of microprocessors. This CPU incorporates a faster clock (100MHz) and 256KB L2 cache. PowerPC 603e is used by Motorola, Inc. under license from IBM.

PowerPC 604™ The third implementation of the PowerPC family of microprocessors currently under development. PowerPC 604 is used by Motorola, Inc. under license from IBM.

PowerPC Reference Platform (PRP) A specification published by the IBM Power Personal Systems Division which defines the devices, interfaces, and data formats that make up a PRP-compliant system using a PowerPC processor.

PowerStack™ RISC PC (System Board) A PowerPC-based computer board platform developed by the Motorola Computer Group. It supports Microsoft's Windows NT and IBM's AIX operating systems.

PRP See PowerPC Reference Platform (PRP).

PRP-compliant See PowerPC Reference Platform (PRP).

PRP Spec See PowerPC Reference Platform (PRP).

PROM **P**rogrammable **R**ead-**O**nly **M**emory

PS/2 **P**ersonal **S**ystem/**2** (IBM)

QFP **Q**uad **F**lat **P**ackage

RAM **R**andom-**A**ccess **M**emory. The temporary memory that a computer uses to hold the instructions and data currently being worked with. All data in RAM is lost when the computer is turned off.

RAS **R**ow **A**ddress **S**trobe. A clock signal used in dynamic RAMs to control the input of the row addresses.

Reduced-Instruction-Set Computer (RISC) A computer in which the processor's instruction set is limited to constant-length instructions that can usually be executed in a single clock cycle.

RFI **R**adio **F**requency **I**nterference

RGB The three separate color signals: **R**ed, **G**reen, and **B**lue. Used with color displays, an interface that uses these three color signals as opposed to an interface used with a monochrome display that requires only a single signal. Both digital and analog RGB interfaces exist.

RISC See Reduced-Instruction-Set Computer (RISC).

ROM **R**ead-**O**nly **M**emory

RTC **R**eal-**T**ime **C**lock

SBC **S**ingle **B**oard **C**omputer

SCSI **S**mall **C**omputer **S**ystems **I**nterface. An industry-standard high-speed interface primarily used for secondary storage. The SCSI-1 implementation provides up to 5 Mbps data transfer.

SCSI-2 (Fast/Wide) An improvement over plain SCSI; and includes command queuing. Fast SCSI provides 10 Mbps data transfer on an 8-bit bus. Wide SCSI provides up to 40 Mbps data transfer on a 16- or 32-bit bus.

serial port A connector that can exchange data with an I/O device one bit at a time. It may operate synchronously or asynchronously, and may include start bits, stop bits, and/or parity.

SIM **S**erial **I**nterface **M**odule

SIMM — **S**ingle **I**nline **M**emory **M**odule. A small circuit board with RAM chips (normally surface mounted) that is designed to fit into a standard slot.

SIO — **S**uper **I/O** controller

SMP — **S**ymmetric **M**ulti**P**rocessing. A computer architecture in which tasks are distributed among two or more local processors.

SMT — **S**urface **M**ount **T**echnology. A method of mounting devices (such as integrated circuits, resistors, capacitors, and others) on a printed circuit board, characterized by not requiring mounting holes. Rather, the devices are soldered to pads on the printed circuit board. Surface-mount devices are typically smaller than the equivalent through-hole devices.

software — The term used to describe any single program or group of programs, languages, operating procedures, and documentation of a computer system. A computing system is normally spoken of as having two major components: hardware and software. Software is the real interface between the user and the computer.

SRAM — **S**tatic **R**andom **A**ccess **M**emory

SSBLT — **S**ource **S**ynchronous **BL**ock **T**ransfer

standard(s) — A set of detailed technical guidelines used as a means of establishing uniformity in an area of hardware or software development.

SVGA — **S**uper **V**ideo **G**raphics **A**rray (IBM). An improved VGA monitor standard that provides at least 256 simultaneous colors and a screen resolution of 800 x 600 pixels.

Teletext — One-way broadcast of digital information. The digital information is injected in the broadcast TV signal, VBI, or full field, The transmission medium could be satellite, microwave, cable, etc. The display medium is a regular TV receiver.

thick Ethernet (10Base5)
An Ethernet implementation in which the physical medium is a double-shielded, 50-ohm coaxial cable capable of carrying data at 10 Mbps for a length of 500 meters (also referred to as thicknet).

thin Ethernet (10Base2)
An Ethernet implementation in which the physical medium is a single-shielded, 50-ohm RG58A/U coaxial cable capable of carrying data at 10 Mbps for a length of 185 meters (also referred to as AUI or thinnet).

twisted-pair Ethernet (10BaseT)
An Ethernet implementation in which the physical medium is an unshielded pair of entwined wires capable of carrying data at 10 Mbps for a maximum distance of 185 meters.

UART **U**niversal **A**synchronous **R**eceiver/**T**ransmitter

UV **U**ltra**V**iolet

UVGA **U**ltra **V**ideo **G**raphics **A**rray. An improved VGA monitor standard that provides at least 256 simultaneous colors and a screen resolution of 1024 x 768 pixels.

Vertical Blanking Interval (VBI)
The time it takes the beam to fly back to the top of the screen in order to retrace the opposite field (odd or even). VBI is on the order of 20 TV lines. Teletext information is transmitted over 4 of these lines (lines 14-17).

VESA (bus) **V**ideo **E**lectronics **S**tandards **A**ssociation (or VL bus). An internal interconnect standard for transferring video information to a computer display system.

VGA **V**ideo **G**raphics **A**rray (IBM). The third and most common monitor standard used today. It provides up to 256 simultaneous colors and a screen resolution of 640 x 480 pixels.

virtual address A binary address issued by a CPU that indirectly refers to the location of information in primary memory, such as main memory. When data is copied from disk to main memory, the physical address is changed to the virtual address.

VL bus — See **V**ESA **L**ocal bus (VL bus).

VMEchip2 — MCG second generation VMEbus interface ASIC (Motorola)

VME2PCI — MCG ASIC that interfaces between the PCI bus and the VMEchip2 device.

volatile memory — A memory in which the data content is lost when the power supply is disconnected.

VRAM — **V**ideo (Dynamic) **R**andom **A**ccess **M**emory. Memory chips with two ports, one used for random accesses and the other capable of serial accesses. Once the serial port has been initialized (with a transfer cycle), it can operate independently of the random port. This frees the random port for CPU accesses. The result of adding the serial port is a significantly reduced amount of interference from screen refresh. VRAMs cost more per bit than DRAMs.

Windows NT™ — The trademark representing **Windows N**ew **T**echnology, a computer operating system developed by the Microsoft Corporation.

XGA — E**X**tended **G**raphics **A**rray. An improved IBM VGA monitor standard that provides at least 256 simultaneous colors and a screen resolution of 1024 x 768 pixels.

Y Signal — Luminance. Parameter that determines the brightness (but not the color) of each spot (pixel) on a CRT screen in color or B/W systems.